ATOM: A general calibration framework for multi-modal, multi-sensor systems

Miguel Oliveira[a,c], Eurico Pedrosa[b,c], André Silva Aguiar[d], Daniela Rato[a,c,*], Filipe Neves dos Santos[d], Paulo Dias[b,c], Vítor Santos[a,c]

[a] *Department of Mechanical Engineering (DEM), University of Aveiro, 3810-193 Aveiro, Portugal*
[b] *Department of Electronics, Telecommunications and Informatics (DETI), University of Aveiro, 3810-193 Aveiro, Portugal*
[c] *Institute of Electronics and Informatics Engineering of Aveiro (IEETA), University of Aveiro, 3810-193 Aveiro, Portugal*
[d] *INESC TEC - INESC Technology and Science; 4200-465, Porto, Portugal*

Abstract

The fusion of data from different sensors often requires that an accurate geometric transformation between the sensors is known. The procedure by which these transformations is estimated is known as sensor calibration. The vast majority of state of the art calibration approaches focus on specific pairwise combinations of sensor modalities, unsuitable to calibrate robotic systems containing multiple sensors of varied modalities. This paper presents a novel calibration methodology which is applicable to multi-sensor, multi-modal robotic systems. The approach formulates the calibration as an extended optimization problem, in which the poses of the calibration patterns are also estimated. It makes use of a topological representation of the coordinates frames in the system, in order to recalculate the poses of the sensors throughout the optimization. Sensor poses are retrieved from the combination of geometric transformations which are atomic, in the sense that they are indivisible. As such, we refer to this approach as ATOM - Atomic Transformations Optimization Method. This makes the approach applicable to different calibration problems, such as sensor to sensor, sensor in motion, or sensor to coordinate frame. Additionally, the proposed approach provides advanced functionalities, integrated into ROS, designed to support the several stages of a complete calibration procedure. Results covering several robotic platforms and a large spectrum of calibration problems show that the methodology is in fact general, and achieves calibrations which are as accurate as the ones provided by state of the art methods designed to operate only for specific combinations of pairwise modalities.

Keywords: Extrinsic calibration, Intrinsic calibration, Registration, Multi-modal, Multi-sensor, ROS

1. Introduction

Whenever an intelligent or robotic system is composed of two or more sensors, a procedure that estimates the geometric transformations between those sensors is required. The process is called extrinsic calibration or sensor registration [1].

The vast majority of sensor fusion techniques operate under the assumption that accurate geometrical transformations between the sensors that collect the data are known. This is valid for many different applications, from the simple design of sensors that collect RGB and depth information [2], to a stereo camera pair designed to carry out underwater 3D reconstruction [3], to more complex sensor setups such as intelligent vehicles [4, 5], smart camera networks [6], or even multi-sensor image analysis from datasets captured by diverse airborne or spaceborne sensors [7]. Thus, one may argue that an accurate estimation of those transformations, i.e., a good extrinsic calibration, is a critical component of any data fusion methodology.

Although the problem of extrinsic calibration is well defined, in practice there are several variants of that problem. As discussed previously, the classical formulation seeks to provide an estimate of the transformations between the several sensors in a system. These will be referred to as **sensor to sensor calibration problems**. One variant is the calibration of a single

*Corresponding author
Email addresses: `mriem@ua.pt` (Miguel Oliveira), `efp@ua.pt` (Eurico Pedrosa), `andre.s.aguiar@inesctec.pt` (André Silva Aguiar), `danielarato@ua.pt` (Daniela Rato), `filipe.n.santos@inesctec.pt` (Filipe Neves dos Santos), `paulo.dias@ua.pt` (Paulo Dias), `vitor@ua.pt` (Vítor Santos)

sensor which moves over time. Here, the goal is to find the geometric transformations between the poses of the sensor at the time each data was collected. For example, Boucher et al. [8], Agarwal et al. [9] propose structure from motion methodologies in which the motion of a camera, i.e. the set of camera poses, is recovered in order to reconstruct the scene. We will refer to these as **sensor in motion calibration problems**. Finally, there is another variant of the extrinsic calibration problem: the calibration of a sensor w.r.t. a particular coordinate frame. In these cases, the sensor must move in order to collect data from different viewpoints requiring, therefore, that the motion of the system must be known. Thus, in these cases, one requires a kinematic chain which can be actuated for the purpose of collecting data from multiple viewpoints. We will refer to these as **sensor to frame calibration problems**. One example of a sensor to frame calibration problem was presented in [10], where the transformation between a 2D Light Detection And Ranging (LiDAR) and the pan and tilt unit in which it was mounted was estimated.

Calibration problems can also be defined based on the number of sensors as well as their modalities. Common sensor modalities include RGB cameras, depth cameras, LiDARs, 3D LiDARs, Radio Detection And Rangings (RaDARs), etc. In addition to this, some robotic systems sometimes have several sensors, which results in a large number of possible system configurations, containing single-modality pairwise configurations (e.g. camera to camera or 2D LiDAR to 2D LiDAR as in [5, 11]), single-modality multi-sensor configurations (e.g. cameras networks as in [6]), multi-modal pairwise configurations (e.g. camera to depth camera as in [2], or camera to 3D LiDAR as in [12]) and, finally, the general scenario of the calibration of multi-modal, multi-sensor systems (depth camera and multiple RGB cameras as in [13], several cameras and inertial measurement units as in [14], calibration of sensors in the PR2 robot as in [15]). Obviously, it is also possible to compound this criteria with the former to get, for example, a single-sensor single modality, sensor to frame calibration problem (e.g. [16]), or a multi-sensor multi-modality problem (e.g. [17]).

The large amount of variants of the extrinsic calibration problem, either due to the configuration of the system, the number of sensors, or the sensor modalities, have led to large efforts from the research community in addressing each and every one of these variants, as will be detailed in section 2. However, these works have focused mainly on tackling the many combinations of pairwise configurations of different modalities. In fact, there are very few works which consider the general case of the calibration of multi-modal, multi-sensor systems, and even less if one considers systems that may have sensor to sensor, sensor in motion and sensor to coordinate frame configurations.

This paper proposes a novel methodology called Atomic Transformations Optimization Method (ATOM) which generalizes the extrinsic calibration procedure in such way that it is able to carry out the calibration of all the cases discussed above. Atomic transformations are geometric transformations which are not aggregated, i.e. they are indivisible. Optimizing these transformations fully generalizes the calibration problem, as will be detailed in section 3.

Robot Operating System (ROS) [18] based architectures are now the golden standard in the development of robotic systems. There are several ROS based calibration packages in the public domain, from (Open Source Computer Vision Library (OpenCV) [19] based intrinsic camera[1] and stereo camera pair[2] calibrations, to RGB-D camera calibration [20][3] [21][4], to hand-eye calibration [22, 23][5]. ROS integration ranges from using input data recorded in *rosbag* files as in [24], to using ROS standard messages as in [22]. Despite this, there is no available ROS package that provides a complete solution for the calibration of robots in general.

In addition to a general calibration methodology, ATOM also offers a complete calibration framework that addresses all the stages of a calibration pipeline: definition of the initial pose of the sensors, data collection and labelling and, finally, the actual optimization procedure. All these tools are seamlessly integrated into ROS.

This document is organized as follows: section 2 will detail existing calibration approaches and how they relate to our proposal, section 3 describes the proposed methodology, section 4 offers a complete description of the framework, available tools and integration with ROS, and finally section 5 and section 6 provide results and conclusions respectively.

2. Related Work

In general terms, an extrinsic calibration requires that two or more sets of data are associated by matching

[1]`http://wiki.ros.org/openni_launch/Tutorials/IntrinsicCalibration`

[2]`http://wiki.ros.org/camera_calibration/Tutorials/StereoCalibration`

[3]`http://alexteichman.com/octo/clams`

[4]`https://github.com/code-iai/iai_kinect2`

[5]`http://wiki.ros.org/rc_visard/Tutorials/HandEyeCalibration`

unique key points in those sets. Once these associations are retrieved, it is possible to formulate a procedure that estimates the parameters of the geometric transformations between the sensors using the associations as input. An extrinsic calibration can be designed using closed form solutions [25, 26, 27, 28, 29, 30, 31] or with iterative procedures [32, 16, 33, 34, 17]. In the later, an optimization problem is formulated with the goal of finding the poses of the sensors, i.e. the configuration of optimized parameters, which minimize the distance between the key-point associations. Since the accuracy of those associations is critical to the estimation procedure, most calibration approaches make use of calibration patterns, i.e., objects that are robustly and accurately detected. Moreover, in the case of multi-modal sensor systems, a calibration pattern adequate to all existing sensor modalities must be selected. For camera sensor modalities, the standard calibration patterns are chessboards [35, 36] and fiducial markers [37, 38, 39]. For range measuring based modalities such as LiDAR or RaDAR, patterns which contain a distinct physical shape signatures are used, such as spherical objects [40, 41, 42, 43], conic objects [44], or planar cardboards containing circular holes [45, 46]. A minority of works perform calibration without using a pattern. In these, the features in the scene are used as input to the calibration. In autonomous driving, for example, lane detection and vanishing point tracking are common approaches for online calibration [47, 48]. [10] propose to make use of planes in indoor scenes (e.g. walls, floor, etc.) for supporting the calibration of a 2D LiDAR on a pan and tilt unit system. Patternless calibration approaches have the advantage of operating continuously if necessary, but loose in accuracy and robustness when compared to offline, one shot procedures. As such, offline calibrations are still the most commonly used.

2.1. Sensor to Sensor calibration problems

As a critical component of intelligent or robotic systems, the topic of extrinsic calibration has been extensively addressed in the literature over the past decades. The large bulk of these works have focused on a particular case of **sensor to sensor calibration problems**, which is the case of systems containing a single pair of sensors, i.e. pairwise calibrations. In this regard, most combinations of modalities have already been covered: RGB to RGB camera calibration [49, 50, 51, 52, 53, 41], RGB to depth camera calibration [43, 54, 55, 56, 57, 58], RGB camera to 2D LiDAR [59, 41, 57, 60, 61, 62, 63, 46], 2D LiDAR to 3D LiDAR [44]; RGB camera to 3D LiDAR [64, 46, 65], RGB camera to radar [45], etc.

The problem of RGB to RGB camera calibration has received a great deal of attention from the research community, in particular in the case of two camera stereo systems [49, 50, 51, 52, 53]. The classic approach is to carry out an optimization which estimates the transformation between the cameras using the reprojection error as guidance. The optimization procedure may also include the estimation of the intrinsic parameters [50]. RGB to RGB camera calibration methodologies have also been proposed to address the problem of online calibration [49, 51] and also markerless calibration [52].

The great majority of the works found in the literature focused on pairwise calibration between an RGB camera and a 3D LiDAR are based on the work of Huang and Barth [12], where the calibration is performed in two stages: first using closed-form equation; and second, a maximum likelihood estimation refinement. Similarly, Verma et al. [66] use a standard chessboard to calibrate a perspective/fisheye camera and a 3D LiDAR using a Genetic Algorithm. Wang et al. [67] 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 and Bonnifait [68], Guindel et al. [46] use circle-based patterns to perform the extrinsic calibration. Mirzaei et al. [69] 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 photo-realistic 3D reconstruction of indoor and outdoor scenes. Pandey et al. [70] 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 and Grizzle [71] use a pattern of known dimensions and geometry and estimates the pattern to LiDAR pose automatically using a fitting algorithm.

Pairwise calibration approaches consider that the sensors are rigidly attached. As such, these approaches cannot handle the cases where a sensor moves during the calibration procedure as in the **sensor in motion calibration problem** and the **sensor to frame calibration problem**.

2.2. Hand-eye calibration problem

Hand-eye calibration is defined as the process of estimating the transformation between the end-effector, i.e.

the *hand* of a robotic arm, and a camera, the *eye*, which is rigidly attached to that end-effector [25]. The formulation of the problem is expressed as:

$$\mathbf{AX} = \mathbf{ZB} \, , \tag{1}$$

where **A** represents the known geometric transformation from the *hand* to the robotic arm *base*, obtained using forward kinematics, **B** denotes the known transformation from the *eye* to the *world object*, **X** specifies the unknown transformation from the robotic arm *base* to the *world object* and **Z** is the unknown transformation from the *hand* to the *eye* [28].

Initial works proposed closed form solutions which tackled the rotation and translation components of (1) separately [72, 28]. Later on, Liz et al. [30] propose a close form method to based on dual quaternions and the Kronecker product that was able to obtain translations and rotations simultaneously. Also using the Kronecker product, Shah [31] introduced a close form method that combines the Kronecker product and single value decomposition to find a simultaneous solution.

In recent years, the formulation in (1) has been tackled using optimization methods to find the unknown transformations, i.e. **X** and **Z**. Tabb and Yousef [32] introduced an iterative method based on the minimization of camera reprojection error, which solves the rotation and translation components simultaneously. The paper compares the proposed approach with other solutions in the literature, and shows that the iterative approach based on the reprojection error offers the best calibration results, which is later confirmed in [16].

One shortcoming of hand-eye calibration approaches is that they are not able to tackle the multi-sensor case. To address this, Tabb and Yousef [32] propose a formulation that bundles all cameras into a single optimization procedure, in what they refer to as hand-eye(s) calibration problem. Another common limitation is that they are not able to handle both the *eye-on-hand* and *eye-to-base* use cases simultaneously.

Because the robotic arm must move in order to collect different views of the calibration pattern and what is sought is the transformation from the camera to the end-effector, the hand-eye calibration is both a sensor in motion calibration problem as well as a sensor to frame calibration problem. However, the focal aspect of hand-eye calibration approaches is that all of them are specialized in this particular problem, which means that they are not suited to address **sensor to sensor calibration problems**, rarely tackle the multi-sensor case, and only take into account a single sensor modality (RGB cameras).

2.3. Extension to the multi-sensor, multi-modal case

The generalization of pairwise calibration approaches to the multi-sensor case, in which the number of sensors is greater than two, is not straightforward. One reason for this is that most calibration approaches are designed to operate by processing the data from a sensor tandem. In this way, the inclusion of a third sensor would require additional pairwise procedures for all pairwise combinations in the system. In fact, the most common solution for the calibration of multi-sensor systems, to which we refer to as sequential pairwise approaches, is to run several pairwise calibrations and arrange them in a graph-like sequential procedure, in which one sensor calibrates with another, that then relates to a third sensor, and so forth. Each pairwise calibration will provide an estimate for the geometric transformation that links two of the sensors in the system. For example, Zhou et al. [73] present 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. [74] 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. Similarly, Kim and Park [75] perform six pairwise calibrations between a 3D LiDAR and six monocular cameras mounted in an hexagonal plate that constitute an omnidirectional camera.

In these sequential pairwise approaches, the complete system can be described by a topological representation where nodes are sensor coordinate systems and the edges are the estimated transformations between those coordinate systems. Providing that the topological representation is not disconnected, it is possible to compute the transformation from any sensor to another by retrieving the topological path between these sensors, and combining the corresponding transformations that have been estimated by pairwise calibrations.

Figure 1 shows an examples of these topological representations considering a system with 4 sensors. Naturally, the structure of this topological representation depends on the pairwise calibrations that were selected. For example, in visual hodometry [76], a sensor in motion calibration problem, that structure is of a linear nature, since each image is connected only to images within its spatio-temporal neighborhood. In the case of intelligent vehicles or mobile robotic platforms in general, one common approach is to establish one sensor as the reference sensor and calibrate all sensors w.r.t. the reference one, which results in a pyramidal topological structure (see Figure 1(a)). One example is [41], in which a methodology for calibrating an intelligent vehicle [77] is proposed, wherein all sensors are paired with a reference sensor.

Sequential pairwise approaches have several shortcomings, which are detailed in the next lines. As discussed previously, they imply that one must choose which pairwise combinations of sensors in the system are used to run pairwise calibrations. Note that there are many other alternative topological configurations to Figure 1(a). Assume, for example, that the RGB to LiDAR calibration is not very accurate: in that case one could replace the calibration from *Sensor 0* to *Sensor 3* by a calibration from *Sensor 2* to *Sensor 3*, which has a different combination of modalities, *depth* and LiDAR. This would result in a different topological configuration for the same set of sensors. To ensure that there is only one transformation from any sensor to any other, the topological structure must be non redundant. That leads to a selection of sensor combinations that is only a subset of the total pairwise combinations that exist in the system. Thus, the first shortcoming is that not all available data is used for the calibration of the system: transformations are estimated using only data provided from the selected sensor tandems, despite the fact that data from additional sensors could be available and their inclusion prove relevant to the overall accuracy of the calibration procedure. Figure 1 shows these unused connections as dashed lines.

It is also possible to consider all available connections, but that leads to the additional problem of how to cope with the redundancy in the topology. Santos et al. [78] propose a multi-pairwise approach which considers a set of pairwise calibrations. This results in a redundant topological structure which is then post-processed by averaging the redundant transformations. Figure 1(b) shows an example of this configuration.

Because the transformations are computed in sequence, it is quite possible that errors may accumulate over the sequence of connections. Since the accumulation effect should be more noticeable for longer sequences, a valid strategy is to reduce the depth of the topological graph as much as possible (e.g. a one level pyramidal structure as in [41]). However, this approach is limited to scenarios where the field of view of all sensors overlap with the field of view of the reference sensor, which is not always the case.

Another disadvantage is that the topology of the transformations in the system is defined by the convenience of the user, rather than because of the constraints or limitations of a calibration procedure. In our view, the need to design a topology that accomodates the sequential arrangement of the calibration procedure is not ideal. For example, in Figure 1(a) the decision to connect *Sensor 0* to *Sensor 3* could be based on the need to avoid the connection between *Sensor 2* to *Sensor 3*, because the calibration of a *depth* to LiDAR modality pair is less accurate or even non-existent. This kind of convenience based decision is very common but, instead, it would be much more interesting to be able to define the structure of transformations that best suits the system, according to more relevant criteria, and then having a calibration procedure that is able to calibrate the system regardless of its topological configuration.

Another problem is scalability, because when using pairwise combinations between sensors, the number of pairwise combinations will grow considerably with the increase in the number of sensors. This is notoriously more problematic when attempting to use redundant topological structures, because the number of paths to go from one sensor to another explodes very quickly. As a result, sequential calibration approaches using sensor pairs do not scale efficiently. Sequential pairwise calibration approaches also do not scale well for multimodal systems. The reason is that, since the approach is designed to evaluate sensor tandems, a specific methodology must be designed for each combination of modalities, e.g. RGB to RGB camera, RGB camera to 3D LiDAR, 3D LiDAR to 3D LiDAR etc. The inclusion of a novel modality brings about the need to develop a new set of methods to evaluate the novel modality against all those previously known. Figure 1(c) provides an example of these implications. The inclusion of a RaDAR sensor would require three novel calibration methodologies to be developed.

There are a few works which address the problem of calibration from a multi-sensor, simultaneous optimization, perspective. Noel et al. [79] 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. [13] propose a joint objective function to simultaneously calibrate three RGB cameras w.r.t. a depth camera. Authors report a significant improvement in the accuracy of the calibration. In Rehder et al. [14], an approach for joint estimation of both temporal offsets and spatial transformations between sensors is presented. This approach is one of the few that

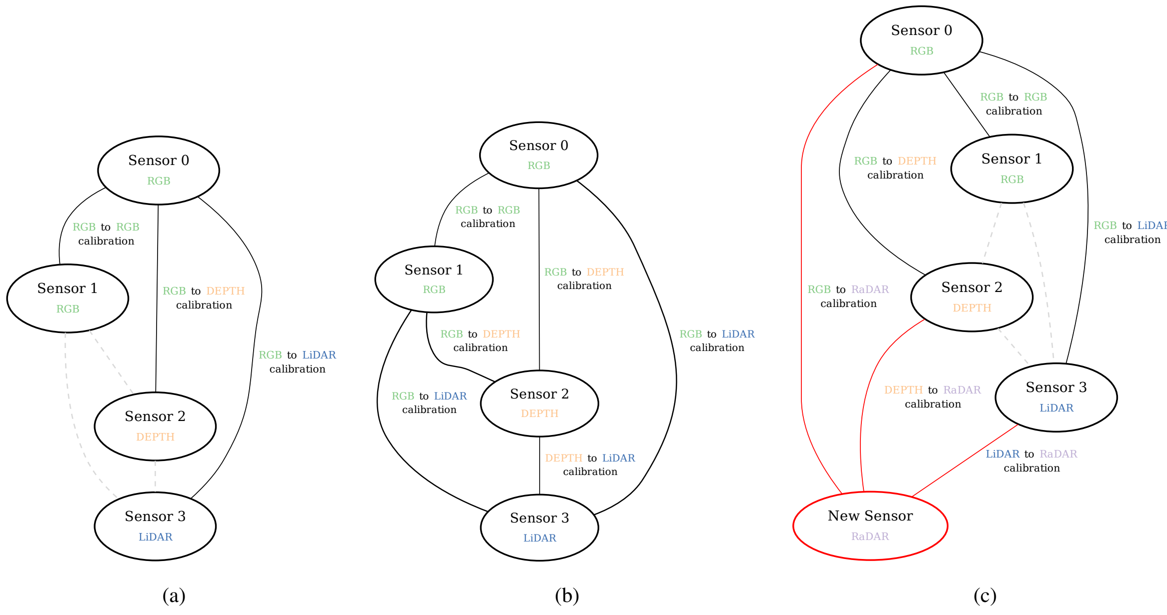

Figure 1: Some alternative topological structures for a calibration procedure: (a) flat pyramidal configuration with Sensor 0 as reference; (b) using all possible pairwise combinations in the calibration; (c) the inclusion of an additional sensor of a previously unknown RaDAR modality. Red lines indicate new pairwise calibrations which would have to be implemented. Sensors are represented by ellipses, with the modality of each sensor indicated below its name; solid lines represent pairwise combinations which are used in the calibration procedure; dashed lines represent additional connections which are not used.

are 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 [15], 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.

This paper presents a general calibration formulation that comprises **sensor to sensor**, **sensor in motion** and **sensor to frame** calibration problems, as will be detailed next. Our previous works have focused on the calibration of intelligent vehicles [34], agricultural robots [80], and hand-eye systems [81]. In all those works, the same baseline method based on the optimization of Atomic Transformations is used.

3. Proposed Approach

As a case study to better illustrate the concepts that will be detailed ahead, we will use a robotic system called Multi-Modal Test roBot (MMTBot) [6]. This is a simulated, conceptual robot, designed to test the performance of advanced calibration methodologies. The system contains the following sensors: *lidar*, a 3D LiDAR mounted on the left side of a tripod; *world camera*, an RGB camera mounted on the right side of that same tripod; and a *hand camera*, a second RGB camera assembled on the effector link of a robotic manipulator, which is mounted on a table. The complete system is displayed in Figure 2.

MMTBot is a multi-modal robotic system, which combines RGB and LiDAR modalities. Moreover, one of the RGB cameras (*hand camera*) is assembled on the end effector of the roboic manipulator, which brings a hand-eye calibration problem into the system. In fact, because MMTBot is simultaneously a multi-sensor, multi-modal, sensor to sensor, as well as a sensor to frame calibration problem, there is no solution in the literature which is able to conduct the complete, simulateneous, calibration of this system.

3.1. Problem Formulation

From the analysis conducted in section 2, it is clear that most calibration approaches operate using a pair-

[6]https://github.com/miguelriemoliveira/mmtbot

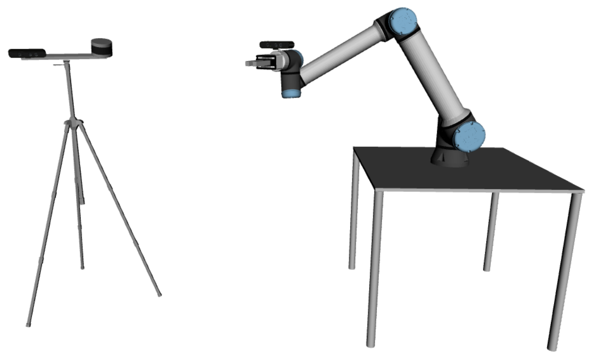
Figure 2: The Multi-Modal Test roBot (MMTBot), a simulated robotic system containing, from left to right in the figure: a RGB camera and a 3D LiDAR mounted on a tripod, and a second RGB camera assembled on the end effector of a robotic manipulator.

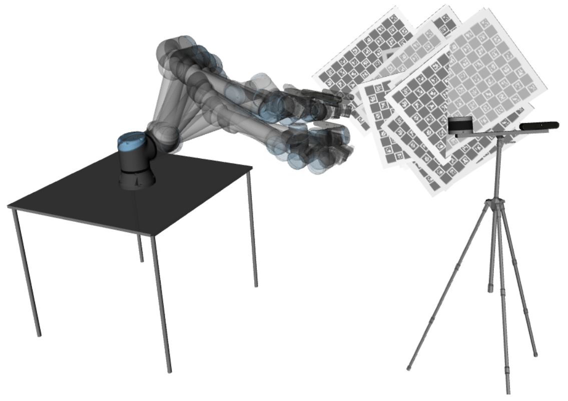
Figure 3: A dataset with multiple collections, each with a different configuration of the robotic arm and pose of the calibration pattern.

wise or set of pairwise evaluations. Thus, a calibration of sensor s_1 w.r.t. a sensor s_2 can be generaly denoted as the minimization of the following expression:

$$\underset{{}^{s_1}\mathbf{T}^{s_2}}{\operatorname{argmin}} \sum_{c \in \mathcal{C}} \sum_{d \in \mathcal{D}} e\left({}^{s_1}\mathbf{T}^{s_2}, d_{[c,s_1]}, d_{[c,s_2]}, \{\lambda_{s_1}\}, \{\lambda_{s_2}\}\right), \tag{2}$$

where ${}^{s_1}\mathbf{T}^{s_2}$ is the 4×4 homogeneous transformation matrix between sensor s_1 and sensor s_2; d is the detection taken from the set of detections $\mathcal{D}$, which are representations of the calibration pattern on data from the sensors; $\{\lambda\}$ denotes the set of additional parameters of the sensor required by the cost or error function $e(\cdot)$. The cost function $e(\cdot)$ outputs one or more scalar values refered to as residuals, which are to be minimized by the optimization procedure; and finally, c denotes the collection taken from the set of collections $\mathcal{C}$. To achieve accurate estimates, calibration procedures minimize a cost computed over several views of the calibration pattern and / or robot pose. We refer to these time stamps in which sensor data is acquired as *collections*. Collections store not only sensor data but also the state of the robotic system. As an example, a collection of the MMTBot includes two images (one from each camera) plus a point cloud (captured by the 3d lidar), as well as all the transformations between coordinate frames. All this data is collected at the same time. Figure 3 shows a set of collections $\mathcal{C}$, which have different configurations of the robotic arm as well as poses of the calibration pattern.

An example of the instantiation of (2) would be the following: assume that both sensors are RGB cameras. Each of the syncronized pair of images acquired by the cameras constitutes a collection c, of two images in this case. Then, the corners of the pattern as detected by a chessboard detector would constitute the set of detections $d_{[c,s]}$ for a given image (collection c) and sensor s. Thus, in this case the detections would be represented as pixel coordinates, i.e., points in $\mathbb{Z}^2$, and the cost function would recompute the distance between the projected pixel coordinates and the detected coordinates, often known as reprojection error. Since the computation of the projection would require the intrinsic parameters of the cameras (and possibly also the distortion parameters) these would be included in the set of additional parameters $\{\lambda\}$.

A multi-sensor scenario is defined as one in which the number of sensors is greater than two. Let $\mathcal{S} : n(\mathcal{S}) \geq 3$ denote the set of sensors in the system, and $\mathbb{S}$ represent the set of pairwise combinations of the elements of $\mathcal{S}$. The extension of this pairwise approach to the multi-sensor case requires that all, or at least a subset, of the pairwise combinations in $\mathbb{S}$ are evaluated, which extends (2) as follows:

$$\underset{\{{}^{s_i}\mathbf{T}^{s_j}\}}{\operatorname{argmin}} \sum_{\{s_i,s_j\} \in \mathbb{S}} \sum_{c \in \mathcal{C}} \sum_{d \in \mathcal{D}} e\left({}^{s_i}\mathbf{T}^{s_j}, d_{[c,s_i]}, d_{[c,s_j]}, \{\lambda_{s_i}\}, \{\lambda_{s_j}\}\right), \tag{3}$$

where $\{{}^{s_i}\mathbf{T}^{s_j}\}$ is the set of estimated transformations that correspond to the set of pairwise combinations $\mathbb{S}$. The critical issue with the pairwise formulation in (3) is observed for multi-modal systems, that is, where the sensors $\mathcal{S}$ in the system have diferent modalities. In these cases, a variant of the cost function $e(\cdot)$ must be implemented to cope with each pair of modalities in the system. If the cost function is symmetric, meaning it will provide the same results regardless of the order by which the modalities are evaluated, the number of function variants to be implemented is computed by

Table 1: Number of cost function variants required as a function of the type of cost function and the number of sensor modalities in the system.

	# modalities					
Type of cost function	2	3	4	5	10	20
Sensor Pairwise Symmetric	1	3	6	10	45	190
Sensor Pairwise Assymmetric	2	6	12	20	90	380
Sensor to Pattern (ATOM)	2	3	4	5	10	20

the number of pairwise combinations of the modalities in the system. If, on the other hand, the cost function is assymmetric, e.g. if evaluating a RGB camera w.r.t. a LiDAR is different from evaluating a LiDAR w.r.t. a RGB camera, then the pairwise permutations of the modalities define the number of cost function variants to be implemented. Table 1 provides an analysis of the problem, and clearly shows the scalability issue inherent to pairwise formulations.

This paper proposes a formulation which is an alternative to parwise approaches. Our proposal is to augment the optimization problem, by extending the set of parameters to optimize in order to include an estimate, for each collection, of the transformations from the world coordinate frame w to the calibration pattern p coordinate frame, denoted as ${}^{w}\mathbf{T}^{p}$. This enables the cost function to be considerably simplified, since it now may use the estimated pose of the pattern to compute the cost in a sensor to pattern logic. As such, the cost function does not have to evaluate pairs of sensors, which avoids all the problems discussed above. The proposed formulation is the following:

$$\underset{\{{}^{s}\mathbf{T}^{w}\},\{{}^{w}\mathbf{T}^{p}\}}{\operatorname{argmin}} \sum_{s\in\mathcal{S}} \sum_{c\in\mathcal{C}} \sum_{d\in\mathcal{D}} e\Big({}^{s}\mathbf{T}^{w}, {}^{w}\mathbf{T}^{p}_{c}, d_{[c,s]}, \{\lambda_s\}\Big), \quad (4)$$

where $\{{}^{s}\mathbf{T}^{w}\}$ labelsize=lst of transformations from the world coordinate frame w to each sensor s, which must be static in order to be calibrated, reason why they do not contain collection index c. On the other hand, the transformation ${}^{w}\mathbf{T}^{p}_{c}$ between the world w and pattern p coordinate frames contains an index c because the pattern is placed on different positions for every collection. Table 1 also shows how the proposed approach scales well with the inclusion of additional modalities.

3.2. Atomic Transformations

The proposed calibration framework also requires the definition of a topological structure which contains information of the relationships between coordinate frames, often referred to as a transformation tree. The transformation tree for MMTBot is presented in Figure 4. There, it is possible to observe that the graph of transformations is a detailed representation of the robotic system. As such, this representation often includes portions of the tree which are not relevant for the calibration procedure, as is the case of the gripper component, which for that reason appears greyed out in Figure 4. In addition to this, the tree also contains both static as well as dynamic transformations, marked as black and blue arrows in Figure 4 respectively. A static transformation is defined as a transformation that does not change for all collections of data used in the calibration, i.e. if $\mathbf{T}_{c_i} = \mathbf{T}_{c_j}, \forall c_i, c_j \in \mathcal{C}$. For example, the transformation from the *table* to the *base* of the robotic arm is static, while the transformation between the *base* and the *shoulder* of that robotic arm is dynamic, since it changes according to the motion of the arm. The transformation tree also displays several branches of variable depth, as can be observed by comparing the two cameras, for example. Figure 4 also includes a pattern coordinate frame, and a transformation from the *world* to the *pattern* denoted as ${}^{w}\mathbf{T}^{p}$ in the previous section.

The vast majority of calibration approaches typically reduce this complexity, with the goal of simplifying the calibration procedure. This is done by computing aggregate transformations or eliminating some coordinate frames. For example, most approaches designed to tackle hand-eye calibration problems make use of (1), in which the transformations **A**, **X**, **Z** and **B** are often aggregated transformations.

We argue that, by preserving the complete topological structure of transformations, it is possible to generalize all calibration problems. The proposal is to include all coordinate systems in the topological representation, and to store the values of all these transformations for all collections, so that the complete transformation tree can be recomputed during the optimization procedure when required, and for any collection. We refer to the transformations stored in this topological representation as atomic transformations, in the sense that they are not aggregated, are indivisible. The notation $\mathcal{T}$ is used to distinguish atomic transformations from the other transformations. The method that we propose uses these atomic transformations to formulate the optimization procedure carried out for the calibration. As such, we refer to it as Atomic Transformations Optimization Method (ATOM).

Having a connected transformation tree, it is possible to retrieve the unique topological route from one point in the graph to another, i.e., the path from any coordinate frame (f_a) to any other (f_b). With this, the transforma-

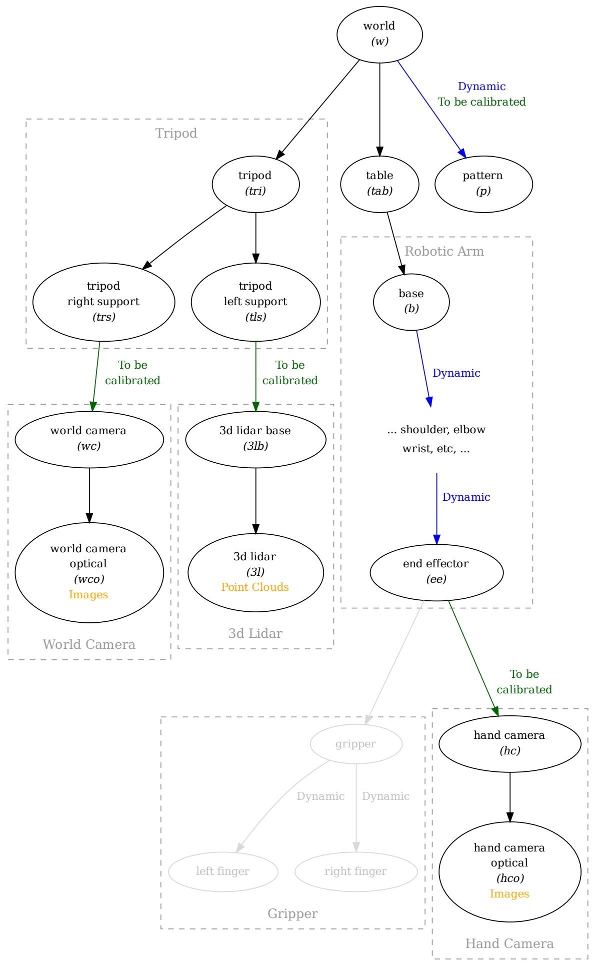

Figure 4: The transformation tree for the Multi-Modal Test roBot (MMTBot), a simulated robot containing two RGB cameras and a lidar. The presented topology follows the structure of the sensors and arm drivers as provided by the manufacturers, although some reference frames have been renamed or ommited for better readability.

tion between frames can be computed as follows:

$$^{f_a}\mathbf{T}_c^{f_b} = \prod_{f_n \in f_a \rightarrowtriangle f_b} {}^{f_n}\mathcal{T}_c^{f_{n+1}} , \qquad (5)$$

where ${}^{f_n}\mathcal{T}_c^{f_{n+1}}$ denotes the atomic transformation from frame f_n to frame f_{n+1}, and $f_a \rightarrowtriangle f_b$ is the topological path that constitutes the sequence of coordinate frames transversed to go from frame f_a to frame f_b. Note that the collection c appears in (5) to account for the fact that some of the transformations may be dynamic, i.e., change from collection to collection.

Since the goal of the cost function is to evaluate the error using a sensor to pattern approach (see (4)), what is sought is the transformation from the coordinate frame of the sensor to the pattern p, denoted as ${}^{s}\mathbf{T}_c^{p}$, which can be retrieved from (5). This expression is derived for all the sensors in the system. Also, to configure the calibration procedure, it is necessary to select, for each sensor, which of the atomic transformations will be estimated during the process. The selection is arbitrary, provided that the selected transformation is static and that it is included in the topological path of the respective sensor. Also, the number of selected transformations is not limited. Take the example of Figure 4, which shows the transformations that have been marked to be calibrated. In the case of the *world camera* sensor, the selected transformation was from the *tripod right support* to the *world camera*. It is possible to, instead calibrate the *world camera* to the *world camera optical* transformation. Moreover, both transformations can be selected simultaneously.

In the example of Figure 4 the selection of transformations to be calibrated is the simplest solution: just a single, the most intuitive transformation per sensor was selected. Sensor *world camera* has a topological path $wco \rightarrowtriangle p = \{wco, wc, trs, tri, w, p\}$, where wco, wc, trs, tri, w and p, stand for *world camera optical*, *world camera*, *tripod right support*, *tripod*, *world* and *pattern*, respectively. Assuming the selections of transformations to be calibrated shown in Figure 4, (5) becomes for *world camera* sensor:

$$^{wco}\mathbf{T}_c^{p} = \prod_{f_n \in wco \rightarrowtriangle wc} {}^{f_n}\mathcal{T}_c^{f_{n+1}} \cdot {}^{wc}\hat{\mathcal{T}}^{trs} \cdot \prod_{f_n \in trs \rightarrowtriangle w} {}^{f_n}\mathcal{T}_c^{f_{n+1}} \cdot {}^{w}\hat{\mathcal{T}}_c^{p} , \quad (6)$$

where the hat notation $\hat{\mathcal{T}}$ is used to signal that these atomic transformations are estimated. Also, note that ${}^{wc}\hat{\mathcal{T}}^{trs}$ does not have a collection index c because it must be static by definition. Sensor *3d lidar* has a topological path $3l \rightarrowtriangle p = \{3l, 3lb, tls, tri, w, p\}$, respectively *3d lidar*, *3d lidar base*, *tripod left support*, *tripod*, *world* and *pattern*. In this case, (5) becomes:

$$^{3l}\mathbf{T}_c^{p} = \prod_{f_n \in 3l \rightarrowtriangle 3lb} {}^{f_n}\mathcal{T}_c^{f_{n+1}} \cdot {}^{3lb}\hat{\mathcal{T}}^{tls} \cdot \prod_{f_n \in tls \rightarrowtriangle w} {}^{f_n}\mathcal{T}_c^{f_{n+1}} \cdot {}^{w}\hat{\mathcal{T}}_c^{p} , \quad (7)$$

and sensor *hand camera* has a topological path $hco \rightarrowtriangle p = \{hco, hc, ee, \ldots, b, tab, w, p\}$, respectively *hand camera optical*, *hand camera*, *end effector*, *base*, *table*, *world* and *pattern*. In this case, (5) becomes:

$$^{hco}\mathbf{T}_c^{p} = \prod_{f_n \in hco \rightarrowtriangle hc} {}^{f_n}\mathcal{T}_c^{f_{n+1}} \cdot {}^{hc}\hat{\mathcal{T}}^{ee} \cdot \prod_{f_n \in ee \rightarrowtriangle w} {}^{f_n}\mathcal{T}_c^{f_{n+1}} \cdot {}^{w}\hat{\mathcal{T}}_c^{p} . \quad (8)$$

Note that (6), (7) and (8) derive the particular expressions of the MMTBot sensors. These expressions are shown for clarity of presentation alone, because it is the expression (5) that is actually implemented, which automatically derives into the particular expression of each sensor. This proves that the proposed formulation successfully generalizes any calibration problem, including sensor to sensor, sensor in motion and sensor to frame calibration problems.

The formulation for extrinsic calibration using a sensor to pattern paradigm, as proposed in (4), can now be rewritten to accommodate the atomic transformations, which results in:

$$\underset{\{\hat{\mathcal{T}}\}}{\operatorname{argmin}} \sum_{s\in\mathcal{S}} \sum_{c\in\mathcal{C}} \sum_{d\in\mathcal{D}} e\left({}^{s}\mathbf{T}_{c}^{p}, d_{[c,s]}, \{\lambda_s\}\right), \qquad (9)$$

where $\{\hat{\mathcal{T}}\}$ denotes the set of atomic transformations marked to be optimized, and ${}^{s}\mathbf{T}_{c}^{p}$ is the transformation from the sensor to the pattern, which is computed using the atomic transformations.

3.3. RGB Camera Error Function

The calibration proposed in this paper is formulated in (9). This is a generic expression, in which the error function $e(\cdot)$ must be instantiated for each existing sensor modality. For the RGB sensor modality we propose to use the reprojection error, computed by projecting the known corners of the calibration pattern ($\mathbf{x}$) to the camera image, and comparing these projected coordinates with the coordinates of the correponding detections in the image (x):

$$e_{[c,s,d]} = \left\| \mathrm{x}_{[c,s,d]} - \mathcal{P}\left(\left[{}^{s}\mathbf{T}_{c}^{p} \cdot \mathbf{x}_{d}\right]_{xyz}, \mathbf{k}_s, \mathbf{u}_s \right) \right\|_F^2, \qquad (10)$$

where $\mathbf{x}_d$ is the three-dimensional homogeneous coordinates of the pattern corner that corresponds to detection d, defined in the pattern's local coordinate frame; the operator $[\cdot]_{xyz}$ is an operator that extracts the x,y and z coordinates (removes the homogeneous coordinate); $\mathbf{k}$ and $\mathbf{u}$ are the vector of camera intrinsics and distortion parameters, both included in the set of additional parameters for the sensor, i.e., $\mathbf{k}_s, \mathbf{u}_s \in \{\lambda_s\}$; $\mathcal{P}$ is the projection function; $\mathrm{x}_{[c,s,d]}$ denotes the two-dimensional pixel coordinates of detection d, found in the image acquired by sensor s at collection c; and finnaly $\|\cdot\|_F^2$ denotes the Frobenius norm.

3.4. 3D LiDAR Error Function

The error function $e(\cdot)$ for 3D LiDAR modality is composed of two distinct evaluations which compute the orthogonal $o(\cdot)$ and logitudinal $l(\cdot)$ errors for each detection, i.e. $e(\cdot) = \{o(\cdot), l(\cdot)\}$. In the context of this modality, a detection d is a range measurement performed by the sensor which is labelled to belong to the calibration pattern, and the set of detections $\mathcal{D}$ is a subset of the point cloud provided by the sensor.

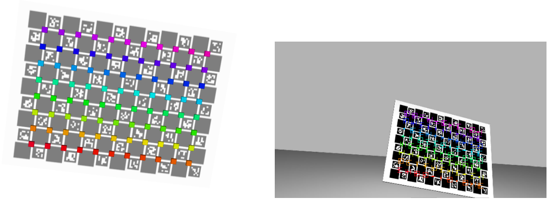
(a) (b)

Figure 5: Projection of calibration pattern's corners (*a*) to the images of *world camera* (*b*). Color coding distinguishes the detections.

The orthogonal error $o(\cdot)$ is computed by evaluating the distance between the estimated pattern XoY plane and the detection coordinates. One simple way to accomplish this is to transform the detection coordinates from the LiDAR's local coordinate frame to the pattern's coordinate frame, and evaluate the z coordinate, as follows:

$$o_{[c,s,d]} = \left[\left({}^{s}\mathbf{T}_{c}^{p}\right)^{-1} \cdot \mathbf{x}_d \right]_z \qquad (11)$$

where $\mathbf{x}_d$ denotes the three dimensional coordinates of the detections in the LiDAR's local coordinate frame, $\left({}^{s}\mathbf{T}_{c}^{p}\right)^{-1}$ is the transformation from the pattern p to the sensor s, and the operator $[\cdot]_z$ extracts the z coordinate of the point.

The longitudinal error $l(\cdot)$ makes use of the 3D coordinates of the boundary of the calibration pattern with measurements labeled as being boundaries in the set of pattern measurements. Let $\mathcal{B} \in \mathcal{D}$ denote the set of measurements labeled as boundaries, and $\mathbf{x}_b$ be the 3D coordinates of the boundary point b. The process by which these points are identified is called labelling and will be detailed in subsequent sections. By transforming $\mathbf{x}_b$ to the pattern's coordinate frame, it is then possible to compare these coordinates with the known geometric structure of the calibration pattern. Let $\mathcal{Q}$ denote the set of 3D coordinates defined in the pattern's local coordinate frame that are obtained by the (spatialy periodic) sampling of the lines that delimit the physical body of the pattern. Thus, the longitudinal error can be written as follows:

$$l_{[c,s,d]} = \min_{\mathbf{q}\in\mathcal{Q}} \left(\left\| \left[\mathbf{q} - \left({}^{s}\mathbf{T}_{c}^{p}\right)^{-1} \cdot \mathbf{x}_b \right]_{xy} \right\|_F^2 \right), \qquad (12)$$

where $\mathbf{q}$ denotes a 3D point defined in the pattern's local coordinate frame and obtained through sampling as detailed above, and the operator $[\cdot]_{xy}$ extracts the x and

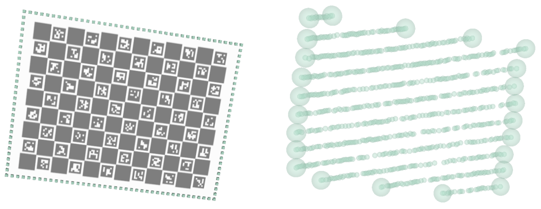
(a) (b)

Figure 6: The physical boundaries of the calibration pattern are represented by a set of sampled points $\mathcal{Q}$ (green cubes in (*a*)). The longitudinal evaluation finds the smallest distance between each of the sensor measurements labeled as boundaries $\mathbf{x}_b$ (large green spheres in (*b*)) and all the points in $\mathcal{Q}$. The orthogonal evaluation measures the distance between the pattern's XoY plane and the sensor measurements labeled as belonging to the pattern, i.e., the detections $\mathbf{x}_d$ (small green spheres in (*b*)).

y coordinates of the point, which account for the longitudinal component of the error. Figure 6 shows the process of computing the LiDAR modality cost function. The small green spheres in Figure 6 (*b*) represent the range measurements labeled as belonging to the pattern, i.e., $\mathbf{x}_d$ in (11)). For the orthogonal evaluation, these are compared against the estimated pattern XoY plane. The large green spheres in Figure 6 (*b*) represent the range measurements labeled as boundaries, i.e. $\mathbf{x}_b$ in (12). In the case of the longitudinal evaluation these points are compared against the points sampled from known model of the calibration pattern, denoted as $\mathbf{q}$ in (12) and represented as the green cubes of Figure 6 (*b*).

We have presented the cost functions for the RGB and 3D LiDAR modalities. Natturally, the inclusion of additional modalities would require the definition of other cost functions, but their integration in the proposed framework is straightforward.

3.5. Normalization of Residuals

ATOM proposes the usage of distinct error functions for diferent sensor modalities. The output of these functions is called residuals. Thus, each function contributes to a global vector of residuals.

Note that, in the case of RGB cameras (see (10)) the error is measured in pixels. On the other hand, errors for LiDAR, defined in (11) and (12) are expressed in meters. Clearly, 1 pixel is not the same as 1 meter but, to the optimizer, these residuals have the exact same magnitude. This mismatch in units may lead to differences in the magnitude of the residuals computed for distinct modalities, which will result in an unbalanced optimization, where larger residuals from specific modalities may overshadow residuals from other modalities.

To address this problem of multi-modal residuals, we propose to compute normalized residuals $\tilde{e}$ using a normalization mechanism as follows:

$$\tilde{e}_{[c,s,d]} = \eta_{m(s)} \cdot e_{[c,s,d]} \tag{13}$$

where $\eta_{m(s)}$ is the normalization factor for a given modality $m(s)$, and $m(\cdot)$ is a function that retrieves the modality of sensor s. The normalization factors are computed before the optimization starts, and are constant throughout the procedure. The normalization factor for a given modality n, denoted as η_n, is computed by taking into account all of the residuals of that same modality:

$$\eta_n = \frac{\sum\limits_{s\in\mathcal{S}} \sum\limits_{c\in\mathcal{C}} \sum\limits_{d\in\mathcal{D}} \delta\left(n, m(s)\right)}{\sum\limits_{s\in\mathcal{S}} \sum\limits_{c\in\mathcal{C}} \sum\limits_{d\in\mathcal{D}} e^0_{[c,s,d]} \cdot \delta(n, m(s))} \tag{14}$$

where $e^0_{[\cdot]}$ denotes the error computed using the values of the parameters as provided at the start of the calibration procedure, i.e., the initial estimate, and $\delta(\cdot)$ is the delta Kronecker function. Note that the normalization factors are constant values during the optimization, and are calculated once with the residuals that result from the initial guess.

The proposed framework is general in the sense that any parameter that is used to compute the error, i.e. that has impact on the residuals, can be estimated. This includes not only the parameters that encode the geometric transformations, but also other additional parameters. The most straightforward example is that of the intrinsic parameters of RGB cameras, which are used to compute the reprojection error (see (10)), and can therefore be simultaneously estimated during the calibration procedure. As such, ATOM is not an exclusively extrinsic calibration methodology. Rather, since intrinsic parameters (and others) may also be included in the optimization, we view it is a general methodology for the calibration of sensors. This leads to the extension of the problem formulation, from the one proposed in (9), to the following:

$$\underset{\{\hat{\mathcal{T}}\},\{\hat{\lambda}\}}{\operatorname{argmin}} \sum_{s\in\mathcal{S}} \sum_{c\in\mathcal{C}} \sum_{d\in\mathcal{D}} \tilde{e}\Big({}^{s}\mathbf{T}^{p}_{c}, d_{[c,s]}, \{\hat{\lambda}_s\}\Big), \tag{15}$$

where $\{\hat{\lambda}\}$ denotes the set of additional parameters that are also being estimated, along with the atomic transformations.

3.6. Non-linear Least Squares Optimization

By definition, *least squares* tries to minimize the squared residuals (i.e the predicted error) of an objective function with respect to its parameters:

$$\underset{\phi}{\operatorname{argmin}} \sum \|\mathcal{F}(\phi)\|^2 \ , \tag{16}$$

where $\mathcal{F}$ is an objective function that returns the residuals vector given the parameters ϕ. Considering (15), we have $\phi = \{\{\hat{\mathcal{T}}\}, \{\hat{\lambda}\}\}$ and $\mathcal{F} = \tilde{e}$. While the additional parameters $\{\hat{\lambda}\}$ are direclty used, the atomic transformations $\{\hat{\mathcal{T}}\}$ are encoded as $t_x, t_y, t_z, r_1, r_2, r_3$, where t are the translation components of the transformation, and r denote the rotation components of that atomic transformation $\hat{\mathcal{T}}$, represented thought the angle-axis parameterization, which is used in order to avoid introducing more sensitivity than the one inherent to the problem itself [82].

Using a *non-linear* approach, the parameters ϕ can be estimated using the following iterative update rule:

$$\phi^{u+1} = \phi^{u} + \Delta\phi \ , \tag{17}$$

where $\Delta\phi$ is the update step calculated by an non-linear least squares algorithm. At each step, the model is linearized by a first-order Taylor expansion about ϕ^u, which results in the following normal equations (in matrix notation):

$$(\mathbf{J}^{\intercal}\mathbf{J})\Delta\phi = \mathbf{J}^{\intercal}\mathcal{F}(\phi^{\mathbf{u}}) \ , \tag{18}$$

where $\mathbf{J}$ is the Jacobian matrix of $\mathcal{F}$ with respect to $\phi^{\mathbf{u}}$. The normal equations are the base for many existing algorithms that solve a non-linear least squares optimization problem.

The calibration is handled as a sparse problem because the optimization parameters do not have influence over all of the residuals. For example, the intrinsic parameters of a given camera sensor only produce a non-zero gradient for the residuals related to that specific camera. Also, the parameters that encode a given atomic transformation only influence the residuals which are computed with a sensor to pattern topological path ($s \rightsquigarrow p$) that includes that transformation. An example from Figure 4: The residuals computed by the *world camera* cost function are not affected by the value of the *end effector* to *hand camera* or the *tripod left support* to *3d lidar base* atomic transformations, as these are not included in the path that goes from frame *hand camera optical* to frame *pattern*. The optimization is solved with the trust region reflective algorithm [83], which is a suitable method for large bounded sparse problems. The Jacobian matrix is calculated with numerical differentiation which, although not the most precise method, provides the necessary flexibility for a general approach.

4. Calibration Framework

A novel approach for conducting the calibration of robotic systems based on the optimization of atomic transformations was proposed in section 3. In that section, the focus was striclty on the formulation of the calibration component as an optimization problem, i.e., the estimation of the geometric transformations between sensors. The complete procedure of calibrating a robotic system accommodates additional stages. In fact, a set of prior steps must be carried out before the actual optimization starts. The work presented in this paper covers all the procedures required to perform a complete calibration of a robotic system. As such, we view ATOM not only as a novel calibration method, but also as a complete calibration framework [7]. With the goal of providing a set of calibration tools that are easily used by the community, ATOM is integrated into ROS, which is the standard library for the development of robotic systems. [18]. There are several ROS based calibration packages available, ranging from intrinsic camera calibration, to stereo extrinsic calibration or even RGB-D camera calibration [8]. There are also some ROS packages dedicated to the hand-eye calibration problem: the Open source Visual Servoing Platform [22, 23] provides a solution here[9], and there is also a commercial solution here[10]. The solutions described already have a significant integration with ROS: in particular, they make use of ROS messages to create a calibration eco-system functioning under the ROS framework. We state an extensive integration with ROS as a key component of the proposed approach. To this end, ATOM contains ROS based tools designed to support not just for the optimization phase, but for all components of the calibration procedure.

As discussed in the previous section, ATOM required a graph of transformations in order to compute the topological path from one coordinate frame to another. For this purpose, ROS uses a tree graph referred to as *tf tree* [84]. In the case of Marchand et al. [22], the system is also integrated with the ROS *tf* library. Many calibration approaches modify the topology of the *tf tree* out of convenience for the calibration procedure. However, this may be problematic, because the *tf tree* is used to support many other functionalities such as robot visualization, colision detection, interactive joint control,

[7]https://github.com/lardemua/atom

[8]https://github.com/code-iai/iai_kinect2

[9]http://wiki.ros.org/visp_hand2eye_calibration

[10]http://wiki.ros.org/rc_visard/Tutorials/HandEyeCalibration

motion planning, etc. All these tools are dependant on the predefined topology of the *tf tree*. If the calibration results in modifications to that topology, those functionalities have to be reconfigured or redesigned, which is often a cumbersome task.

Because ATOM is based on the optimization of atomic transformations, only the values of (some of) the transformations in the tree are modified, not its topology. To take the topology of the system into account, the cost functions recompute, for each function call, the transformation between the sensor to the pattern coordinate frames, i.e., ${}^{s}\mathbf{T}_{c}^{p}$ in (9) is always recalculated using (5). Therefore, a change in one atomic transformation in the chain affects the global sensor pose, and consequently, the error to minimize. The optimization may target multiple transformations of each sensor chain, and is agnostic to whether the remaining links are static or dynamic, since all existing partial transformations are stored for each data collection. This ensures not only that the reestimated values of selected atomic transformations are taken into account in the computation of the error function, but also that (some) transformations in the *tf tree* may be dynamic. This is one important aspect as to why ATOM successfully generalizes **sensor to sensor**, **sensor in motion** and **sensor to frame** calibration problems.

To the best of our knowledge, our approach is one of few which maintains the structure of the transformation graph before and after optimization, and we consider this 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 problems.

Figure 7 provides a schematic of the proposed calibration framework. To perform a system calibration, ATOM requires data logged from the system's sensors, provided in the format of a ROS bag file, and also a description of the configuration of the system, as given in ROS *xacro*[11] or *urdf*[12] formats. Note that these two requirements are external to ATOM and are not considered a heavy burden, since that the typical configuration of robotic systems in ROS already includes them. In ATOM, the calibration procedure is structured in four phases, which occur in sequence: 1) calibration configuration, 2) initial positioning of the sensors, 3) data collection and labelling and 4) calibration. Each of these will be described in detail in the next sections.

[11]http://wiki.ros.org/xacro
[12]http://wiki.ros.org/urdf

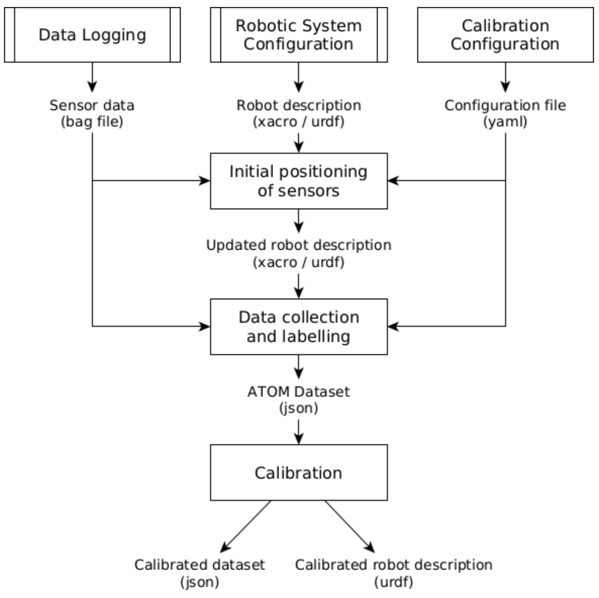
Data Logging
Robotic System Configuration
Calibration Configuration
Sensor data (bag file)
Robot description (xacro / urdf)
Configuration file (yaml)
Initial positioning of sensors
Updated robot description (xacro / urdf)
Data collection and labelling
ATOM Dataset (json)
Calibration
Calibrated dataset (json)
Calibrated robot description (urdf)

Figure 7: The software architecture for the ATOM calibration framework is composed of 4 key components: configuration of the calibration, setting of the initial estimate of the sensor poses, collecting and labelling of data, and the actual calibration procedure.

4.1. 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 was detailed in section 3, in particular the usage of atomic transformations. These transformations are combined through the use of the topological information contained in a transformation tree, which is generated in ROS from the information from an *urdf* or *xacro* file. For the purpose of calibrating the system, additional information must be provided to define which atomic transformations, will be optimized during the calibration procedure. Also, a description of the calibration pattern must be provided to ensure a correct detection and labelling. All this information is defined in a calibration configuration file. An example of the MMTBot calibration file is provided here[13], and a demonstration video here[14]

4.2. Initial positioning of sensors

Since the calibration procedure is formulated as an optimization problem, the initial values of the parameters to be estimated are determinant to the outcome

[13]https://jsonformatter.org/yaml-viewer/a4d145
[14]https://youtu.be/2RTCUt2cdJY

of the optimization. In fact, when the initial parameter values are far from the optimal parameter configuration (the solution), it is possible that the optimization converges into a local minima, thus failing to find adequate values for the parameters. This problem is tackled by ensuring that the initial values contain a plausible first guess. As discussed in section 3, there are several types of parameters to be estimated (see (15)). Different types of parameters require distinct initialization strategies, which will be detailed in the following lines.

The goal of a calibration is to find the position of the sensors, which in our approach is accomplished through the estimation of atomic transformations, e.g. ${}^{wc}\hat{\mathcal{T}}^{trs}$ in (6), ${}^{3lb}\hat{\mathcal{T}}^{tls}$ in (7) or ${}^{hc}\hat{\mathcal{T}}^{ee}$ in (8) for sensors *world camera*, *3d lidar* and *hand camera*, respectively. Hence, the first type of parameters to initialize are these atomic transformations which account for the position of the sensors. To accomplish this, ATOM provides and interactive tool which parses the calibration file and creates a 6-DOF interactive marker associated with each sensor, which overlays on top of the ROS based robot visualization in 3D Visualization Tool for ROS (RVIZ). The sensors are positioned by dragging the interactive markers, which is a simple method to easily generate plausible first guesses for the poses of the sensors. The process is very intuitive because 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 spatially arranged. For example, despite not knowing the exact metric value of the distance between the *world camera* and the *3d lidar*, the user will know that both are more or less at the same height and not more than a meter away from each other. These non-metric, symbolic spatial relationships, in which humans are very proficient, are very useful to generate plausible first guesses. Moreover, the system provides several other visual hints that make the positioning of the sensors an intuitive process. One of these is the ability to visualize how well the data from several sensors aligns. Figure 8 shows an example. On the left side of the figure, an inaccurate positioning of the LiDAR sensor (a) leads to a misaligned projection of the blue spheres on the images (c) and (d). Conversely, an adequate positioning of that same sensor produces a good alignment between the blue spheres in (b) and the patterns in the images (see (e) and (f)). Also, it is possible to see that the alignment between the LiDAR measurements and the physical objects (e.g. table and manipulator) is much better in (b) when compared to (a). The bottom row of Figure 8 shows the alignment between two point clouds, the first produced by the *lidar* and the second by a RGB-D hand camera (which is used here just for the sake of example). Here, it is also very intuitive to realize that the alignment between point clouds is much better when the camera is adequately positioned, the alignment in (h) is better than in (g).

This video[15] shows an example of setting the initial pose of the sensors using the ATOM functionalities. In addition to this, we also provide an example of setting the initial estimate for an intelligent vehicle[16].

As discussed in section 3, the proposed approach is to extend the optimization problem not only to include those transformations that we seek to estimate through the calibration (discussed in the previous paragraph), but also the transformations from the *world* to the *pattern* coordinate frames, denoted as ${}^{w}\hat{\mathcal{T}}_{c}^{p}$ in (6), (7) and (8). Note that the pose of the calibration pattern p is different for every collection c. That means that the pattern moved around in the scenario while the data was being collected, which makes it difficult for any user to keep track of the pose of the pattern over time. Since the user cannot be of assistance in this case, it is not viable to initialize the poses of the calibration patterns using the same methodology as before. To initialize these particular transformations, the following expressions is used:

$$ {}^{w}\hat{\mathcal{T}}_{c}^{p} = \prod_{f_n \in w \rightarrow\!\triangleright s} {}^{f_n}\mathcal{T}_{c}^{f_{n+1}} \cdot g\left(\{\mathrm{x}_{[c,s,d]}\}, \mathbf{k}_s, \mathbf{u}_s\right), \qquad (19) $$

where $g(\cdot)$ denotes a function that solves the perspective-N-point problem [85, 86], given a set of detected pattern corners in the image $(\{\mathrm{x}_{[c,s,d]}\})$ and the intrinsic parameters of the camera sensor s, $\mathbf{k}_s$, $\mathbf{u}_s$, and returns the homogeneous transformation matrix from sensor s to the pattern p, and $w \rightarrow\!\triangleright s$ is the topological path from the world w to the sensor s coordinate frames, which of course must have the RGB modality. If there is more than one RGB sensor in the system, one is arbitrarily selected to produce the initial estimate.

The final type of parameters are those referred to as additional parameters, denoted as $\{\hat{\lambda}\}$ in (15). In the case of RGB camera modalities, these additional parameters are the intrinsic and distortion camera parameters, $\mathbf{k}_s$ and $\mathbf{u}_s$ which are initialized by running a prior intrinsic camera calibration procedure. Naturally, additional parameters of different modalities will require specific initialization strategies.

4.3. Data collection

The proposed optimization mechanism for achieving the calibration of robotic systems was described in sec-

[15]`https://youtu.be/oJLKTqUtZvQ`
[16]`https://youtu.be/zyQF7Goclro`

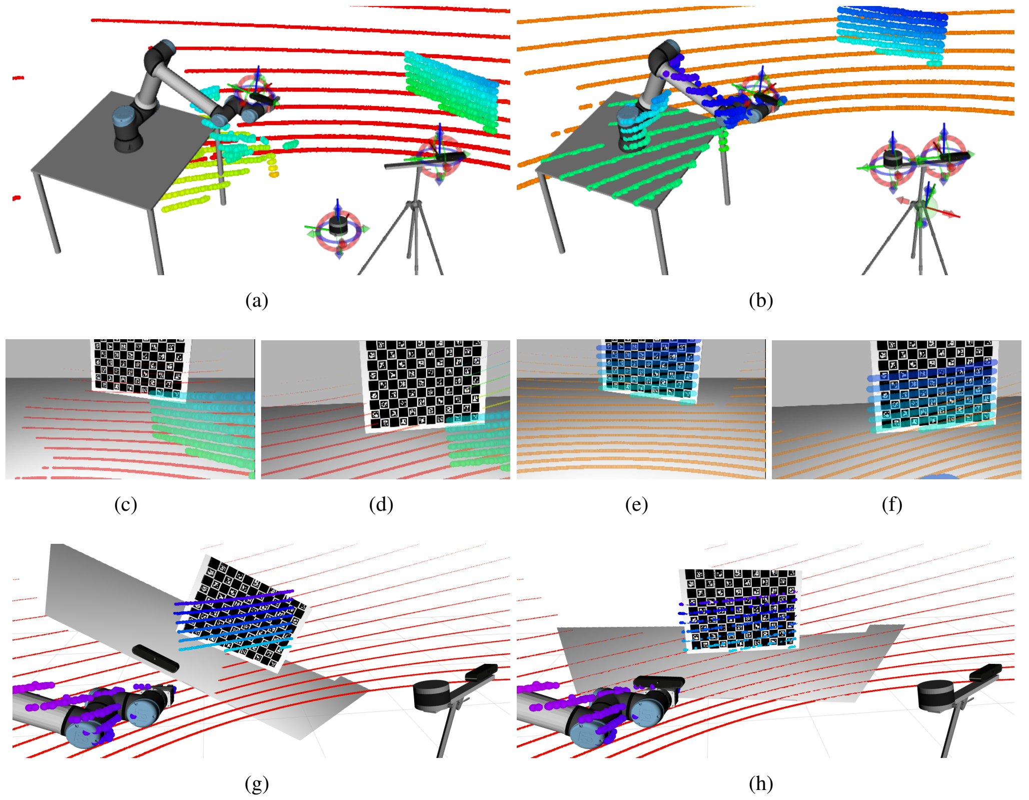

Figure 8: Setting an initial estimate for the sensor poses using RVIZ interactive markers. Top two rows: Inaccurate (left side) vs accurate (right side) positioning of the *3d lidar* sensor. On the left side, the 3D points from the pattern (the blue green spheres in (a)) do not align well with the pattern the images of the *world camera* (c) and the *hand camera* (d) images. The right side shows how an accurate positioning of the sensor aligns the projection of the 3D LiDAR measurements (blue spheres in (b)) with the pattern in the images of the *world camera* (e) and the *hand camera* (f). Bottom row: inaccurate (left) vs accurate (right) positioning of an RGB-D camera. In (g), the point cloud produced by the RGB-D camera (points textured with real colors) is misaligned with LiDAR point cloud (red to blue spheres). In (h), a correct positioning of the RGB-D camera produces a good alignment between point clouds.

tion 3. To ensure that the residuals that are being computed from the current state of estimated parameters are representative, and thus that the calibration is accurate, several views of the calibration pattern should be used. That is the reason why (15) considers the set of collections $\mathcal{C}$. A *collection* is a syncronized snapshot of the data from all sensors in the system at a given time defined by the user. The term is used to denote that, for multi-modal robotic systems, the data collected for each sensor may be different, e.g., images, point clouds, etc. This is a distinction from uni-modal systems: for example, in intrinsic RGB camera calibration, several images (instead of collections) of different views of the calibration pattern are taken to produce the intrinsic and distortion parameters. Each collection also contains the values of all atomic transformations recorded at the corresponding time, so that is possible to recompute the transformations between any two frames (see (5)).

In the case of systems containing more than one sensor, it is very common that the data coming from the sensors is streamed at different frequencies. As such, a syncronization mechanism is required to ensure that the information contained in the collection consistent. This is not a trivial problem to address because the data is not syncronized, i.e., there is never an instant in which all sensors collect data.

This problem is solved through the use of a methodology that ensures the synchronization, assuming the scene has remained static for a *long enough period of time*. In static scenes, the problem of data desynchronization is not observable, which warrants the assumption that for each captured collection the sensor data is *adequately* synchronized. Assuming it is possible to establish an upper bound to the maximum time difference between any data streaming from the robotic system, it is possible to assume that all data messages are synchronized, if the scene has remained static for a period longer than that upper bound. Thus, the methodology establishes that it is the responsibility of the user to ensure that the scene has remained static for a given time period, before triggering the saving of a collection.

There are other approaches which use a similar methodology, in particular by holding the pattern with a tripod (which ensures it does not move) before collecting each image. This approach is used in multi sensor calibration frameworks [14, 24], and also in RGB-D camera calibration procedures[17].

We refer to the set of collections obtained from a given robotic system as an ATOM dataset. It contains a copy of the calibration configuration file, and high level information about each sensor, such as the sensor topological transformation chain, extracted from the transformation tree. In addition, there is also specific information for each collection, i.e., sensor data and labels, as well as values of atomic transformations. It is important to note that the set of collections should contain a sufficiently varied set of pattern poses. As such, collections should preferably have different distances and orientations w.r.t. the calibration pattern, so that the calibration returns more accurate results. Also, if the robotic system contains moving components, the dataset should include several poses of the system. Empirically we have found that 20 to 30 collections is a sufficient number to achieve accurate calibrations. One example of a dataset file for the MMTBot can be found here[18].

4.4. Data labelling

The labelling of data refers to the annotation of the portions of data which view the calibration pattern. The labelling procedure is executed for the data of each sensor, so that all collections have the labels that correspond to the raw sensor data. Labelling can be automatic, semi-automatic or even manual in some cases. The information that is stored in a given label depends on the modality of the sensor which is being labelled.

RGB modality labels consist of the pixel coordinates of the pattern corners detected in the image, and are labelled using one of the many available image-based chessboard detectors [36]. Our system is also compatible with charuco boards, which have the advantage of being detected even if they are partially occluded [37]. Also in this case we make use of off-the-shelf detectors, e.g., [38, 39].

[17]https://github.com/code-iai/iai_kinect2

[18]https://jsonformatter.org/json-parser/a06452

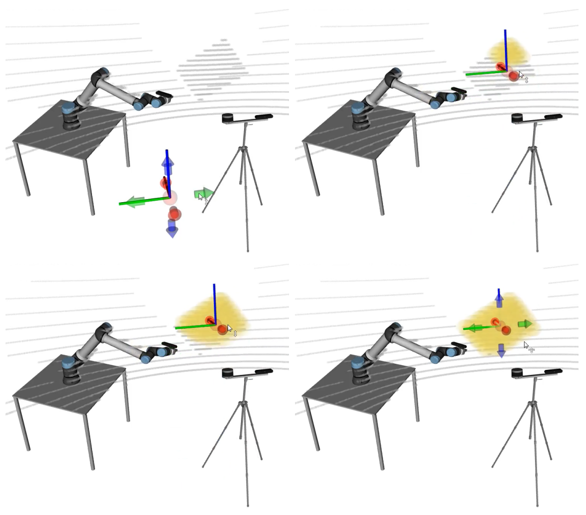

Figure 9: Semi-automatic LiDAR labelling procedure. In a span of about 5 seconds, the user drops the interactive marker close to the pattern, and the system starts tracking it. Yellow spheres indicate *lidar* measurements considered as pattern detections.

The structure of the labels is more complex in the case of the LiDAR modality. As discussed in subsection 3.4, two different types cost functions are used: orthogonal and longitudinal. The range measurements that belong to the pattern is refered to as the set of detections $\mathcal{D}$, and the 3D point coordinates of each are denoted in (11) as $\mathbf{x}_d, \forall d \in \mathcal{D}$. The *lidar* directly produces 3D point coordinates, so it is straightforward to obtain the 3D coordinates $\mathbf{x}_d$ of a given detection d. The difficulty lies in finding the detections $\mathcal{D}$ that belong to the pattern, which is a subset of the complete set of LiDAR measurements. We propose to achieve this using a semi-automatic approach. The intervention of the user is required to set a 3D point which is used as the seed of a region-growing algorithm. Then, starting at the seed point, and assuming that the pattern is physically separated from the other objects in the scene, the algorithm searches in the set of *lidar* measurements for *close enough* points, and includes these into the set of pattern detections $\mathcal{D}$ which are used as seed points in the next iteration. The process is repeated until no propagation occurs. The set of points transversed in the search constitutes the set of detections $\mathcal{D}$. Figure 9 shows a representation of this process.

The second component of the LiDAR cost function is the longitudinal evaluation. For this, it is necessary to retrieve the set of measurements labeled as boundaries, which we denote as $\mathcal{B} \in \mathcal{D}$, in order to recover the 3D coordinates of these points, i.e., $\mathbf{x}_b$ in (12). Once again, the challenge is not to find the 3D coordinates

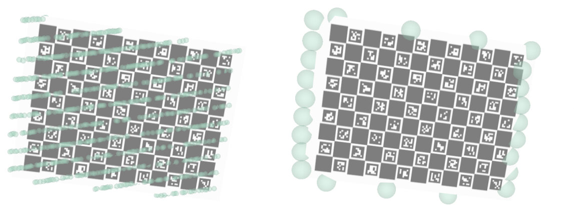
(a) (b)

Figure 10: Labelling of 3D LiDAR data: (a) lidar points labelled as detections; (b) lidar points labelled as boundary points.

but rather which points, from the set $\mathcal{D}$, are boundary points, meaning they are measuring the physical boundaries of the pattern board. The solution for this is to use a spherical parameterization to represent the point coordinates of $\mathbf{x}_d \forall d \in \mathcal{D}$. Let $s_d = [\rho_d, \theta_d, \varphi_d]$ represent the spherical coordinates of the detection d. A 3D lidar generaly contains a much larger horizontal angular resolution, i.e., for the θ angle, when compared with the vertical angular resolution, which corresponds to the φ angle. In fact, 3D *lidars* are said to have scan *layers*, where the vertical angle is the same for a set of measurements spanning all available horizontal angles. The set of measurements corresponding to a scan layer with angle φ_l, denoted as $\{s_{[d,\varphi_l]}\}$, are extracted as follows:

$$\{s_{[d,\varphi_l]}\} = \left\{ s_d : \varphi_l - \frac{\Delta\varphi}{2} \leq \varphi_d < \varphi_l + \frac{\Delta\varphi}{2} \right\}, \forall d \in \mathcal{D}, \tag{20}$$

where $\Delta\varphi$ is the vertical angular resolution of the *lidar*. For each set of detections in a layer, the left and right boundary points, denoted as $\{s_{[b,\varphi_l]}\}$, are extracted by searching for the measurements that have the smallest and largest value of horizontal angle θ:

$$\begin{aligned} \{s_{[b,\varphi_l]}\} = \{s_{[d,\varphi_l]} : \theta_{[d,\varphi_l]} = \max(\{\theta_{[d,\varphi_l]}\}) \vee \\ \theta_{[d,\varphi_l]} = \min(\{\theta_{[d,\varphi_l]}\})\}, \forall d \in \mathcal{D}, \end{aligned} \tag{21}$$

and finally, the complete list of boundary points $\mathcal{B}$ of the calibration pattern, is given by putting together the boundary points detected for each individual scan layer.

Figure 10 shows an example of the LiDAR labelling procedure. The labelled boundary points are signaled by the large green spheres in (b). A video example of the process of producing an ATOM dataset for MMTBot is provided here[19]. This labelling mechanism also works well for 2D LiDARs, as shown in this example of an autonomous vehicle[20].

4.5. Calibration

The vast majority of calibration approaches do little more than printing some information on screen to display the results of the calibration. ATOM provides a great deal of visual feedback, not only at the end of the calibration but also throughout the procedure.

To accomplish this goal, the information stored in an ATOM dataset is published into the ROS ecosystem. The system's configurations for all collections are simultaneously conveyed to ROS, as if they had occurred all at the same time which, not being the case, is the framework under which the calibration procedure operates. 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 and those subtrees are connected so that it is possible to display them toghether.

Figure 11 shows an example of the visualization of a calibration procedure. It is possible to simultaneously visualize the collections that are being used in the calibration (top row), or to select which collections are displayed (bottom row). The ground truth poses of the sensor appear in tranparent mode. These are known only because MMTBot is a simulated robotic system. For real robotic systems there is no ground truth information (the transparent mode is used to show the initial pose of the sensors). It is possible to observe that, as the optimization progresses (from left to right in Figure 11), the sensors move towards the ground truth pose (transparent mode), which means that the optimization is converging towards the optimal solution.

The integration with ROS provides straightforward access to many other interesting functionalities. For example, it is possible to visualize images with the reprojected pattern corners, to display the robot meshes, the position of the reference frames, etc. A complete calibration procedure for MMTBot is displayed here[21].

5. Results

In section 1, three distinct calibration problems were identified: sensor to sensor, sensor in motion and sensor to frame calibration problems. The analysis done in section 2 has shown how the vast majority of the state of the art is focused on one of those problems. Then, in section 3, we described how the proposed approach is able to generalized all calibration problems into a single, unified famework that optimizes atomic transformations.

[19]`www.youtube.com/watch?v=eII_ptyMq5E`
[20]`https://youtu.be/9pGXShLIEHw`
[21]`https://www.youtube.com/watch?v=4B3X_NsX89M`

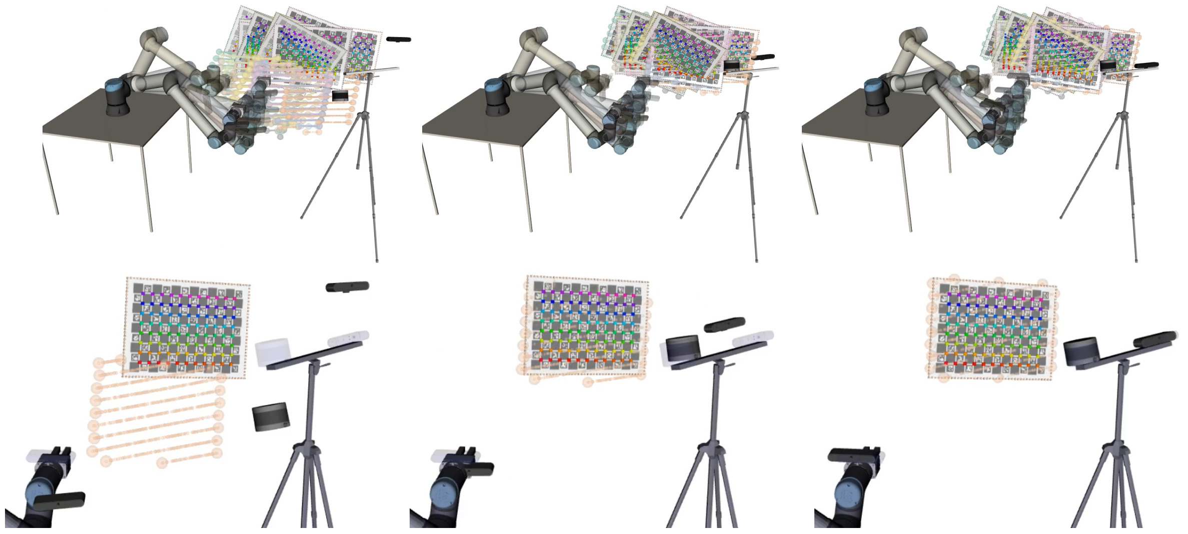

Figure 11: Visualizing the evolution of a calibration procedure from the initial estimate (left) to the solution (right). Top row shows all collections used in this calibration (different color for each collection). Bottom row shows a single collection and, in transparent mode, ground truth poses of the sensors (known because this dataset was generated in simulation).

To show how ATOM may be applied to distinct calibration problems, this section presents results spanning the calibration of four distinct robotic systems, as listed in Table 2. The first is the MMTBot, which was presented before since it was used as a case study in section 3 and section 4. The other three are real robots as displayed in Figure 12: the AtlasCar2 [34, 41] is an autonomous vehicle designed to collect multi-sensor data from real road scenarios (see Figure 12, top); the AgRob V16 is a robotic platform designed for agricultural scenarios in particular steep slope terraced vineyards (see Figure 12, middle); and the Iris UR10e is an experimental system assembled to test several variants of the hand-eye calibration problem (see Figure 12, bottom).

These systems cover all of the calibration problems discussed above. Also, the number of sensors and their modalities differ from system to system. The AtlasCar2 platform represents a sensor to sensor calibration. The AgRob V16 provides additional modalities since it contains a stereo camera and a 3D LiDAR. Finally, the Iris UR10e represents a hand-eye calibration which is both a sensor in motion as well as a sensor to coordinate frame calibration problem. We use three hand-eye problem variants with the Iris UR10e: eye-in-hand, where the camera is assembled on the end effector; eye-on-base, where the camera is assembled rigidly to the arm's base, and a third variant we refer to as joint-hand-base, where both previous cases are combined simultaneously.

The goal is to compare the performance of ATOM with other calibration methods. However, as discussed

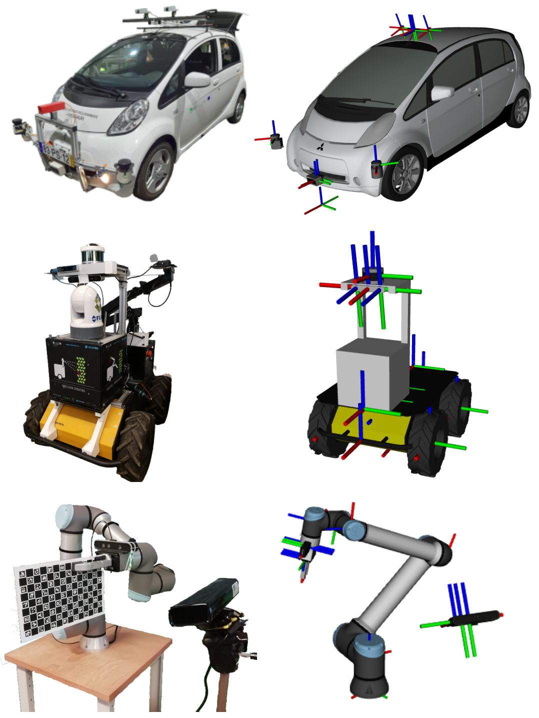

Figure 12: The three non simulated robotic systems used for evaluating the proposed approach: (top) an intelligent vehicle, the AtlasCar2; (middle) an agriculture AGV, the AgRob V16; (bottom) an eye in hand calibration setup, the Iris UR10e. Real image of the robots (left), and the corresponding ROS models (right).

Table 2: Description of the **robotic systems** used to train and test ATOM as well as the **datasets** used for train and evaluation.

System	**Description**	**Sensors**	**Dataset**	Details
MMTBot[(a)]	Simulation with robotic arm All calibration problems	RGB cameras (2x) 3D LiDAR	mmtbot-1 mmtbot-2	29 collections, 9 partial 47 collections, 25 incomplete, 16 partial
AtlasCar2[(b)]	Autonomous vehicle Sensor to sensor	RGB cameras (3x) 2D LiDARs (2x)	atlascar-1 atlascar-2	39 collections, 32 incomplete, 0 partial 21 collections, 18 incomplete, 0 partial
AgRob V16[(c)]	AGV for agriculture Sensor to Sensor	Stereo camera 3D LiDAR	agrob-1 agrob-2 agrob-3	42 collections, 5 incomplete, 38 partial 56 collections, 24 partial 15 collections, 8 partial
Iris ur10e[(d)]	Hand-eye system Sensor in motion, sensor to frame	RGB cameras (x2)	iris-1 iris-2	34 collections, all partial 26 collections, all partial

[(a)] `https://github.com/miguelriemoliveira/mmtbot`
[(b)] `https://github.com/lardemua/atlascar2`
[(c)] `https://github.com/aaguiar96/agrob`
[(c)] `https://github.com/iris-ua/iris_ur10e`

in section 3, there are very few works which are able to calibrate complete robotic systems with multiple sensors and modalities. Rather, the large majority of the calibration approaches focuses on specific pairwise combinations of modalities. Because of this, it is not possible to make a direct comparison between ATOM and other approaches. To address this we propose to conduct a series of pairwise evaluations, which cover the sensors used in the presented systems. In this way we are able to compare against other methodologies, provided they are able to calibrate the selected sensor tandem. Note that, while for other approaches we calibrate only the selected pair of sensors, all results reported for ATOM were collected after a complete system calibration.

The next subsections will cover first the metrics used for evaluating the performance of the calibrations. Then, performance evaluations for distinct combinations of modalities are reported for all robotic systems. Finally, to assess the robustness of the proposed methodology, studies on the influence of several parameters are presented.

5.1. RGB to RGB camera evaluation

To evaluate the RGB-to-RGB camera calibration results three metrics are used: the mean rotation error (in radians), the mean translation error (in meters) and the reprojection error (in pixels).

The mean rotation and translation errors, ε_R and ε_t respectively, measure the difference in the pattern's pose as seen from each RGB camera. These errors are estimated through the evaluating how similar two transformations are. These transformations are obtained through the multiplication of the atomic transformations from the world to the pattern, i.e. $w \multimap p$. In the example of Figure 4 a direct route exists between these two coordinate frames because this transformation is directly estimated by the calibration procedure. In order to obtain trustworthy results the calibration should be done using one dataset, and tested using a different one. As a consequence, the direct route $w \multimap p$ does not exist for evaluation datasets, since these have not been calibrated. However, it is possible to obtain transformation between the world and the pattern coordinate frames (${}^{w}\mathbf{T}_{s}^{p}$), as follows:

$$ {}^{w}\mathbf{T}_{s}^{p} = \prod_{f_n \in w \multimap s} {}^{f_n}\mathcal{T}^{f_{n+1}} \cdot {}^{s}\mathbf{T}^{p} \,, \tag{22}$$

where the index s in ${}^{w}\mathbf{T}_{s}^{p}$ denotes that the transformation is computing using sensor s, and ${}^{s}\mathbf{T}^{p}$ is the transformation between the sensor an the pattern, as estimated through the perspective-N-point [85, 86]. Since two camera sensors, e.g., s_a and s_b, are used in this evaluation, it is possible to obtain the two transformations ${}^{w}\mathbf{T}_{s_a}^{p}$ and ${}^{w}\mathbf{T}_{s_b}^{p}$. The ε_R and ε_t metrics are based on measuring the difference between these transformations, since they should be the same by definition:

$$ {}^{w}\mathbf{T}_{s_a}^{p} \triangleq {}^{w}\mathbf{T}_{s_b}^{p}, \tag{23}$$

which can be rewritten to show the rotation and translation components:

$$ \begin{bmatrix} {}^{w}\mathbf{R}^{s_a} & {}^{w}\mathbf{t}^{s_a} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{s_a}\mathbf{R}^{p} & {}^{s_a}\mathbf{t}^{p} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} {}^{w}\mathbf{R}^{s_b} & {}^{w}\mathbf{t}^{s_b} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{s_b}\mathbf{R}^{p} & {}^{s_b}\mathbf{t}^{p} \\ 0 & 1 \end{bmatrix}, \tag{24}$$

where $\mathbf{R}$ is the rotation matrix and $\mathbf{t}$ is the translation vector. Now, we can define the difference in rotation $\Delta\mathbf{R}$:

$$ \Delta\mathbf{R} = \left({}^{w}\mathbf{R}^{s_a} \cdot {}^{s_a}\mathbf{R}^{p}\right)^{-1} \cdot {}^{w}\mathbf{R}^{s_b} \cdot {}^{s_b}\mathbf{R}^{p} \tag{25}$$

and the difference in translation $\Delta\mathbf{t}$:

$$\Delta\mathbf{t} = {}^{w}\mathbf{R}^{s_a} \cdot {}^{s_a}\mathbf{t}^{p} + {}^{w}\mathbf{t}^{s_a} - {}^{w}\mathbf{R}^{s_b} \cdot {}^{s_b}\mathbf{t}^{p} + {}^{w}\mathbf{t}^{s_b} \ , \tag{26}$$

respectively. Finally, for n states captured, we can now define the mean rotation error as:

$$\varepsilon_{\mathbf{R}} = \frac{1}{n} \sum_{i} \| angle(\Delta\mathbf{R}_i) \| \ , \tag{27}$$

where $angle(\cdot)$ is the angle-axis representation of the rotation. The mean translation error is defined as:

$$\varepsilon_{\mathbf{t}} = \frac{1}{n} \sum_{i} \| \Delta\mathbf{t}_i \| \ . \tag{28}$$

The reprojection error metric is defined by the root-mean-squared error ε_{rms} of the difference between two sets of detected pattern corners. The first set corresponds to the projection of the detected corners in the image of the camera s_a into the image of camera s_b, and the second set corresponds to the detected pattern corners in the image of camera s_b. The projection of the 3D coordinates of the corners of the pattern $\mathbf{x}_d$ onto the image of sensor s is expressed as:

$$\mathrm{x}_d = \mathbf{K}_s \cdot {}^{s}\mathbf{T}^{p} \cdot \mathbf{x}_d, \tag{29}$$

where x denotes the projected 2D image coordinates of the pattern's corners, and $\mathbf{K}_s$ is the intrinsic matrix of camera s. Since the pattern corners are defined in the local pattern's reference frame, they all lie in the $z = 0$ plane. Knowing this, we can simplify (29) by removing the z related components of (29)

$$\mathrm{x}_d = \mathbf{K}_s \cdot [{}^{s}\mathbf{T}^{p}]_{z=0} \cdot \mathbf{x}'_d, \tag{30}$$

where $\mathbf{x}'_d$ denotes the 3D coordinates without the z component, i.e. $\mathbf{x}'_d = [x, y, 1]^{\intercal}$, and the operator $[\cdot]_{z=0}$ extracts the components of the transformation which are not related to z, i.e.:

$${}^{s}\mathbf{M}^{p} = [{}^{s}\mathbf{T}^{p}]_{z=0} = \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{{}^{s}\mathbf{T}^{p}} , \tag{31}$$

which is obtained by removing the z rotation component and the homogeneization components, i.e., removing the highlighted parts of the matrix in (31). Since $\mathbf{x}'_d$ are the same for both cameras s_a and s_b, the following equality can be written:

$$\mathrm{x}_{[d,s_b]} = \mathbf{K}_{s_b} \cdot {}^{s_b}\mathbf{M}^{p} \cdot \left({}^{s_a}\mathbf{M}^{p}\right)^{-1} \cdot \left(\mathbf{K}_{s_a}\right)^{-1} \cdot \mathrm{x}_{[d,s_a]} \ , \tag{32}$$

Table 3: Performance comparison of methods for **RGB to RGB camera** evaluation. Best values highlighted in bold.

Dataset	Method	Sensor pair	$\varepsilon_{\mathbf{R}}$ (rad)	$\varepsilon_{\mathbf{t}}$ (m)	ε_{rms} (pix)
mmtbot-1	OpenCV	*world camera* to	[(a)]	[(a)]	[(a)]
	ATOM	*hand camera*	0.001	0.001	**0.531**
mmtbot-2	OpenCV	*world camera* to	[(a)]	[(a)]	[(a)]
	ATOM	*hand camera*	0.001	0.001	**0.305**
atlascar-1	OpenCV	*left camera* to	0.018	0.013	**1.157**
	ATOM	*right camera*	0.027	0.032	1.198
	OpenCV	*right camera* to	0.244	0.081	3.336
	ATOM	*center camera*	0.096	0.074	**2.375**
	OpenCV	*center camera* to	0.078	0.490	**3.000**
	ATOM	*left camera*	0.090	0.045	3.283
agrob-1	OpenCV	*world camera* to	[(b)]	[(b)]	[(b)]
	ATOM	*hand camera*	0.008	0.003	**0.974**
agrob-2	OpenCV	*world camera* to	0.010	0.006	**0.863**
	ATOM	*hand camera*	0.008	0.005	0.974
ur10e-1	OpenCV	*world camera* to	[(c)]	[(c)]	[(c)]
	ATOM	*hand camera*	0.015	0.002	**1.093**
ur10e-2	OpenCV	*world camera* to	[(c)]	[(c)]	[(c)]
	ATOM	*hand camera*	0.009	0.001	**0.843**

[(a)] OpenCV cannot be used because the hand camera is not static.
[(b)] OpenCV cannot be used because the dataset contains partial detections.
[(c)] OpenCV cannot be used because of both [(a)] and [(b)].

which provides the relationship between the pixel coordinates of the pattern corners in both camera images. The formulation in (32) makes use of the camera to pattern transformation for both cameras, which in turn requires the estimation of the camera to pattern using the perspective-N-point estimation (see (22)). To minimize the error produced by the pnp estimation, we use the procedure only once for one camera, for example, s_a, and then compute the sensor to pattern transformation for the other camera using:

$${}^{s_b}\mathbf{T}^{p} = \left({}^{s_a}\mathbf{T}^{s_b}\right)^{-1} \cdot {}^{s_a}\mathbf{T}^{p} \ , \tag{33}$$

were ${}^{s_a}\mathbf{T}^{s_b}$ is the transformations between the two cameras. Finally, the root-mean-squared error is given by

$$\varepsilon_{rms} = \sqrt{\frac{1}{n(\mathcal{D})} \sum_{d \in \mathcal{D}} \left\| \mathrm{u}_{[d,s_b]} - \mathrm{x}_{[d,s_b]} \right\|_F^2}, \tag{34}$$

where $\mathrm{u}_{[d,s_b]}$ denotes the detected pixel coordinates of the pattern's corners in the image of camera s_b.

The results are presented in Table 3, where the performance of ATOM is compared against the stereo calibration algorithm from OpenCV. Not all datasets can be calibrated using this method, since it mandates that the detections are complete (all corners of the pattern must be detected), and also that the cameras do

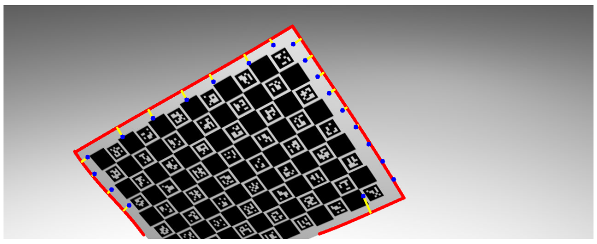
Figure 13: 3D LiDAR to camera reprojection error metric calculation. In the annotation procedure, the four sides of the pattern limits are labelled by selecting multiple points in the image for each side. Then, each side is approximated by a polynomial function as represented by the red curves. The LiDAR reprojected points (blue dots) are then used to calculate the reprojection error (the yellow lines represent the error for each projected point).

Table 4: Performance comparison of methods for **LiDAR to RGB camera** evaluation. Best values highlighted in bold.

Dataset	**Method**	**Sensor pair**	ε_{rms} (pix)
mmtbot-1	ATOM	*3D lidar* to *hand camera*	4.466
		3D lidar to *world camera*	**3.108**
mmtbot-2	ATOM	*3D lidar* to *hand camera*	**4.439**
		3D lidar to *world camera*	6.284
agrob-1	ATOM pairwise	*3D lidar* to *right camera*	3.869
		3D lidar to *left camera*	4.101
	ATOM	*3D lidar* to *right camera*	**3.811**
		3D lidar to *left camera*	3.942
agrob-2	ATOM pairwise	*3D lidar* to *right camera*	6.715
		3D lidar to *left camera*	**6.432**
	ATOM	*3D lidar* to *right camera*	6.537
		3D lidar to *left camera*	6.765

not move w.r.t. each other. Results demonstrate that ATOM achieves similar accuracies when compared to OpenCV's method, despite the fact that ATOM is calibrating the complete systems (all the sensors simultaneously). Also, ATOM accurately calibrates datasets which cannot be tackled by OpenCV's method.

5.2. 3D LiDAR to RGB camera evaluation

To evaluate the calibration of 3D LiDARs and RGB cameras, we propose to use a reprojection error metric that evaluates the error between the labelled boundaries of the pattern and the projection of the LiDAR boundary points onto the RGB camera image. To perform this evaluation, we propose a semi-automatic procedure, where the boundaries of the pattern are manually labelled in the image. This process is divided in three main steps: labelling of the pixels belonging to the pattern limits; reprojection of the pattern's boundary points from the LiDAR's coordinate frame to the image; and calculation of the reprojection error between the labelled and projected points.

The annotation of the pattern's boundaries in the image i.e., the definition of the set of labelled boundary points $\mathcal{V}$, is performed manually by clicking on the image on several boundary points. Then, the four boundaries of the rectagular pattern are aproximated to a polynomial function represented by the red curves in Figure 13, which is later sampled to produce the $\mathrm{v}_b \in \mathcal{V}$ 2D image coordinates. The reason why a polynomial function is used is that a linear regression is not suitable to fit each pattern side since that image has distortion which transforms straight lines into curves.

Now let $\mathbf{x}_b$ denote the 3D coordinates of a boundary point in the pattern's local coordinate frame (see subsection 3.4). The reprojection error ε_{rms} between a LiDAR s_a and a RGB camera s_b is computed as:

$$\varepsilon_{rms} = \sqrt{\frac{1}{n(\mathcal{B})} \sum_{b \in \mathcal{B}} \min_{\mathrm{v}_b in \mathcal{V}} \left(\left\| \mathrm{v}_b - \mathbf{K}_{s_b} \cdot {}^{s}\mathbf{T}^{p} \cdot \mathbf{x}_b \right\|_F^2 \right)} , \tag{35}$$

which finds, for each projected point, the smallest distance to all labelled points. Figure 13 shows the projected points as blue dots, and the yellow lines show the minimum distance to the labelled boundaries found for each projected point.

The LiDAR to RGB camera evaluation was performed using the two robotic platforms in Figure 12 that contain LiDARs: MMTBot and AgRob V16. In the case of MMTBot a complete system calibration was performed, and two evaluations are presented: 3DLiDAR to *hand camera* and 3DLiDAR to *world camera*. In the case of AgRob V16, and in order to evaluate the impact of calibrating the entire system simultaneously, a the complete system calibration is compared with pairwise calibrations, in which only the sensors that are evaluated were calibrated. We refer to these calibrations as ATOM pairwise, and use them to calibrate the *right camera* and *left camera* with the 3DLiDAR separately. Two evaluations are presented: 3DLiDAR to *left camera* and 3dLiDAR to *world camera*.

Table 4 summarizes the results obtained in these experiments. The first observation is that the overall magnitude of the reprojection errors is higher when compared with the reprojection error of RGB to RGB camera evaluations (see Table 3). This is not an inconsistency, since these two reprojections cannot be directly compared. In fact, LiDARs have a smaller resolution when compared with RGB cameras, which may explain why the LiDAR to RGB camera reprojection errors are higher.

Table 5: Impact of the number of partial detections on the performance of ATOM - **RGB to RGB** camera evaluation. Best values highlighted in bold.

Dataset	Sensor pair	# Partial	$\varepsilon_\mathbf{R}$ (rad)	$\varepsilon_\mathbf{t}$ (m)	ε_{rms} (pix)
mmtbot-1	*hand camera* to *world camera*	0 of 10	0.001	0.002	0.374
		1 of 11	0.001	0.002	0.388
		3 of 13	0.001	0.002	0.353
		6 of 16	0.001	0.002	0.379
		9 of 19	0.001	0.002	**0.310**

Results for AgRob V16 show a very similar accuracy between ATOM and ATOM pairwise, presenting only marginal reprojection error differences. This proves that ATOM is adequate to perform full system calibrations without significant loss in the accuracy of the procedure. As mentioned before, this is a big advantage since, for high-dimensional robotic platforms with many sensors, the number of combinations between sensors can be very high an the procedure of sequential pairwise calibration tedious and error prone. Thus, having a framework that is able to calibrate the entire system simultaneously with the same accuracy is very useful.

5.3. Impact of the number of partial detections

As discussed in section 4, the ATOM calibration framework is able to use either chessboard [36] or charuco [37, 38] calibration patterns. In the case of the later, it is possible to detect the pattern even when it is partially occluded [39], which results in a *partial detection* of the calibration pattern. Considering this, it is interesting to assess how the presence of partial detections may impact the accuracy of the calibration. To this end, an experiment was conducted in which a baseline dataset containing 10 collections with no partial detections is augmented with an increasing number of collections containing partial detections. In this experiment, only the MMTBot is used.

Table 5 shows the results using the RGB to RGB cameras evaluation metrics discussed in subsection 5.1. Results show that the partial detections have no impact of the quality of the calibration. Reprojection errors (ε_{rms}) are bellow 0.4 pixels in all cases, which may suggest that ATOM is robust to the presence of partial detections in the datasets.

Table 6 shows the results using the LiDAR to RGB camera evaluation metrics. In this case, the accuracy of the calibration decreases as more partial detections are included, from 4.4 to 5.2 pixels in mmtbot-1, and from 3.4 to 4.4 in the case of mmtbot-2. Thus, it seems that the LiDAR modality is more sensitive to the presence of partial detections. This may be related to the design

Table 6: Impact of the number of partial detections on the performance of ATOM - **LiDAR to RGB camera** evaluation. Best values highlighted in bold.

Dataset	Sensor pair	# Partial	ε_{rms} (pix)
mmtbot-1	*3D lidar* to *hand camera*	0 of 10	**4.443**
		1 of 11	4.701
		3 of 13	4.735
		6 of 14	4.956
		9 of 19	5.234
	3D lidar to *world camera*	0 of 10	**3.490**
		1 of 11	3.713
		3 of 13	3.866
		6 of 16	4.162
		9 of 19	4.435

of the cost functions for this modality, or with the lower resolution of range sensors when compared with image sensors. Nonetheless, note that, in the worst case scenario, 9 of 19 collections are partial, which makes this a challenging dataset, which is tackled with a small loss of accuracy (approximately 1 pixel decrease).

5.4. Impact of the number of incomplete collections

As discussed in section 3, ATOM makes use of a calibration pattern to sensor paradigm in order to design the cost functions. This leads to an optimization framework which is less intricate, where the cost function for each sensor is independant from the others. As a consequence, when the calibration pattern is not detected by a particular sensor, it is still possible to compute, for that collection, the errors associated with the other sensors. We refer these collections where at least one of the sensors did not detect the pattern as *incomplete collections*.

To evaluate how the presence of incomplete collections may affect the accuracy of the calibration produced by ATOM, an experiment was conducted where a baseline dataset containing 10 collections is gradually augmented with incomplete collections.

Table 7 shows the impact on the RGB to RGB cameras evaluation metrics. In the case of the mmtbot-1 datasets, the number of incomplete collections does not appear to affect the accuracy of the calibration, which is consistently bellow a very small value of 0.5 pixels. The datasets from agrob-1 have a lower overall accuracy of approximately 1 or 2 pixels. In this case, we also cannot observe evidence that incomplete collections significantly disrupt the calibration, which suggests, as expected, that ATOM copes well with the presence of incomplete collections.

Table 8 shows the results using the LiDAR to RGB camera metric. Once again, we cannot observe a clear

Table 7: Impact of the number of incomplete collections on the performance of ATOM - **RGB to RGB** camera evaluation. Best values highlighted in bold.

Dataset	**Sensor pair**	**# Incomplete**	$\varepsilon_{\mathbf{R}}$ (rad)	$\varepsilon_{\mathbf{t}}$ (m)	ε_{rms} (pix)
mmtbot-1	*hand camera* to *world camera*	0 of 10	0.001	0.002	0.439
		2 of 12	0.001	0.001	0.429
		3 of 13	0.001	0.001	0.400
		5 of 15	0.001	0.001	**0.390**
agrob-1	*right camera* to *left camera*	0 of 10	0.011	0.011	1.457
		2 of 12	0.006	0.003	**1.031**
		4 of 13	0.009	0.006	2.017
		5 of 15	0.019	0.005	1.076

Table 8: Impact of the number of incomplete collections on the performance of ATOM - **LiDAR to RGB camera** evaluation. Best values highlighted in bold.

Dataset	**Sensor pair**	**# Incomplete**	ε_{rms} (pix)
mmtbot-1	*3D lidar* to *hand camera*	0 of 10	4.057
		2 of 12	4.067
		3 of 13	4.021
		5 of 15	**4.010**
	3D lidar to *world camera*	0 of 10	3.289
		2 of 12	3.282
		3 of 13	**3.200**
		5 of 15	3.218
agrob-1	*3D lidar* to *right camera*	0 of 10	5.496
		2 of 12	**3.920**
		4 of 13	3.964
		5 of 15	4.069
	3D lidar to *left camera*	0 of 10	5.576
		2 of 12	3.930
		4 of 13	**3.849**
		5 of 15	4.169

tendency either way, and the fluctuations are mostly under 1 pixel for all datasets. This reinforces the notion that ATOM is able to tackle the calibrations of datasets containing incomplete collections. Conversely, pairwise approaches required that both sensors detect the calibration pattern or else the collection must be discarded.

5.5. Impact of the quality of the initial estimate

As discussed in section 4, the proposed framework uses an interactive approach that enables the user to set the pose of the sensor. This is used as the initial estimate of the sensor poses, i.e., as the initial values of the parameters to be optimized. Since ATOM uses an optimization mechanism to carry out the calibration, the question of how sensitive the methodology is to the quality of the initial estimates is relevant. Taking into account that the initial parameter estimates are provided by hand, the methodology should be robust enough to handle less accurate estimates.

To address this, we carried out a study of the impact of the quality of the initial estimate on the accuracy of the calibration (Figure 14 (a), (b) and (c)). Quality of the initial estimate is measured using translation (Δd) and rotation ($\Delta\alpha$) deviations from the correct sensor pose, which we know in the case of the MMTBot system since it is simulated. Figure 14 (a) and (b) show the errors of the *hand camera* and *world camera* sensors, respectively. From these, its possible to observe that the sensitivity to rotation is higher when compared with the translation. This is expected, since rotation errors are know to be more critical than translation errors. Also, it is important to note that the final optimization error (e) is very small for a large range of deviations from the correct initial estimate. In fact, both figures show a flat region of very small error within the ranges $0 \leq \Delta d \leq 0.7$ and $0 \leq \Delta\alpha \leq 20$. In other words, ATOM converges to an optimal solution even with deviations of 0.7 meters and 20 degrees. Figure 14 (c) shows the impact of the quality of the initial estimate on the accuracy of the calibration using the LiDAR to RGB camera evaluation metric. When compared with the RGB to RGB camera evaluations we can observe that the LiDAR modality has a larger baseline error, which is consistent with the observations of Table 3 and Table 4. Despite this, we can also see that whithin the ranges of $0 \leq \Delta d \leq 0.7$ and $0 \leq \Delta\alpha \leq 15$ ATOM converges to a minimal error solution. These results show a high degree of robustness to deviations in the initial estimate. Also, we may reasonably expect a human initial estimate to consistently fall bellow 0.7 meters and 15 degrees, which validates the proposal of using an interactive approach to produce the initial estimates. Thus, the conclusion is that ATOM, despite using an optimization mechanism, is sufficiently robust to the quality of the initial estimate for sensor poses.

Figure 14 (d) shows the impact of the quality of the initial estimate to the time it takes to complete the optimization procedure. As expected, deviations from the correct initial estimate will require more effort from the optimized and thus more time to complete the procedure. This is once again notoriously more sensitive to rotation deviations. Nonetheless, in all cases, the maximum time to complete the calibration of the MMTBot robotic system is under 200 seconds. Considering that the calibration is a one shot procedure, this is not relevant.

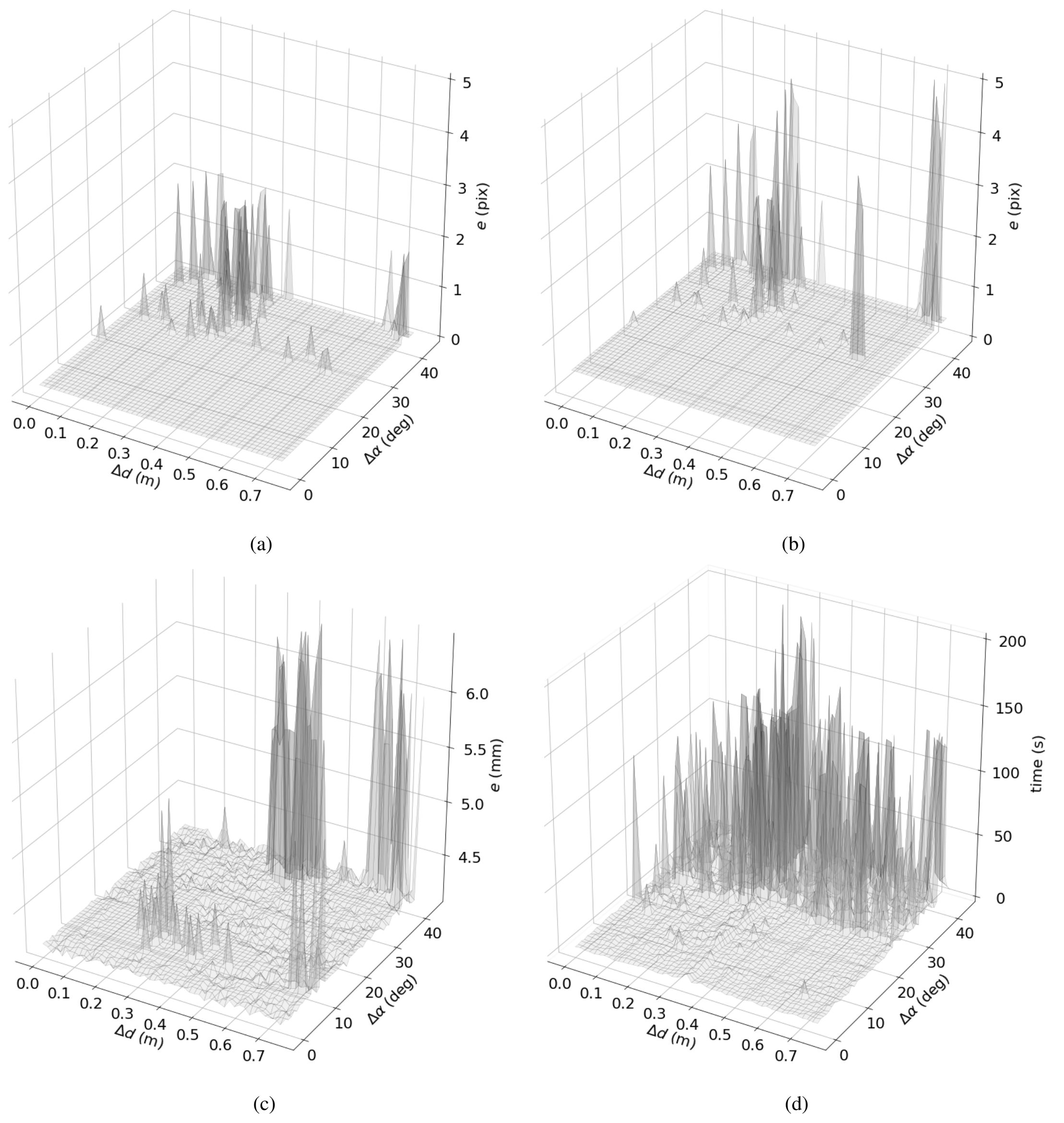

Figure 14: Impact of the quality of the initial sensor pose estimate to the performance of the calibration. Quality of estimate is measured by the deviation from the correct value of translation (Δd) and rotation ($\Delta\alpha$). Errors (e) provived by the cost functions of the sensors: (a) *hand camera*, (b) *world camera*, and (c) *3d lidar*; (d) shows the total optimization time as a function of the initial estimate.

6. Conclusions

This paper proposed a novel general calibration methodology based on the optimization of atomic transformations. Atomic transformations are geometric transformations that are indivisible, i.e., not aggregated, and the advantage of using them is that the problem formulation is suited to cope with the three distinct calibration problems: sensor to sensor, sensor in motion and sensor to frame.

The calibration is formulated as an extended optimization problem, in which the pose of the calibration patterns is also included. Although this formulation augments the problem, since additional parameters are included, it simplifies the definition of cost functions since these are written using the poses of a single sensor and the pattern as input. This is a large advantage over traditional sequential pairwise approaches, since the method is easily scalable to complex robotic systems.

The methodology is general, which makes it very flexible: it can handle any number of sensors of multiple modalities; it handles non static sensors, as in the case of the hand-eye calibration; it does not require that all sensors view the calibration pattern simultaneously, which opens the door to the calibration of systems in which the field of view of the sensors does not entirely overlap. Moreover, all these cases may occur simultaneously.

ATOM is also a calibration framework[22], in the sense that software tools are offered for all the different stages of the calibration procedure. The system is well integrated with ROS, and supports advanced visualization functionalities which are uncommon in most calibration systems.

Results covered four robotic systems with several combinations in the number of sensors and their modalities. These show that the proposed approach is able to calibrate several robotic systems, and that it achieves similar levels of accuracy when compared to other methods, designed to operate only for the sensor tandem that is being evaluated. The robustness of the method is also analysed through several studies that assess the impact of the number of partial detections, incomplete collections, and the initial estimate for the sensor poses.

Future work should address the inclusion of additional sensor modalities, which should be straightforward, since the overall structure of the problem is well defined. We are currently working on the integration of range cameras and 2D LiDARs. In addition to this, we aim to test the methodology and the framework in other robotic systems.

Acknowledgements

This work was supported by the Foundation for Science and Technology, in the context of project UIDB/00127/2020. It was also partially funded by the Project Augmented Humanity [POCI-01-0247-FEDER-046103], P2020, Competitiveness and Internationalization Operational Program, the Lisbon Regional Operational Program, and by the European Regional Development Fund.

References

[1] K. Kodagoda, A. Alempijevic, J. Underwood, S. Kumar, and G. Dissanayake. Sensor registration and calibration using moving targets. In *2006 9th Int. Conf. on Control, Automation, Robotics and Vision*, pages 1–6, 2006. doi: 10.1109/ICARCV.2006.345361.

[2] T. Hanning, A. Lasaruk, and T. Tatschke. Calibration and low-level data fusion algorithms for a parallel 2d/3d-camera. *Information Fusion*, 12(1):37 – 47, 2011. ISSN 1566-2535. doi: 10.1016/j.inffus.2010.01.006. Special Issue on Intelligent Transportation Systems.

[3] A. Pinto and A. Matos. Maresye: A hybrid imaging system for underwater robotic applications. *Information Fusion*, 55:16 – 29, 2020. ISSN 1566-2535. doi: 10.1016/j.inffus.2019.07.014.

[4] M. Tsogas, N. Floudas, P. Lytrivis, A. Amditis, and A. Polychronopoulos. Combined lane and road attributes extraction by fusing data from digital map, laser scanner and camera. *Information Fusion*, 12(1):28 – 36, 2011. ISSN 1566-2535. doi: 10.1016/j.inffus.2010.01.005. Special Issue on Intelligent Transportation Systems.

[5] M. Oliveira, V. Santos, and A. Sappa. Multimodal inverse perspective mapping. *Information Fusion*, 24:108 – 121, 2015. ISSN 1566-2535. doi: 10.1016/j.inffus.2014.09.003.

[6] W. Jiuqing, C. Xu, B. Shaocong, and L. Li. Distributed data association in smart camera network via dual decomposition. *Information Fusion*, 39:120 – 138, 2018. ISSN 1566-2535. doi: 10.1016/j.inffus.2017.04.007.

[7] B. Rasti and P. Ghamisi. Remote sensing image classification using subspace sensor fusion. *Information Fusion*, 64:121 – 130, 2020. ISSN 1566-2535. doi: 10.1016/j.inffus.2020.07.002.

[8] C. Boucher, J. Noyer, and M. Benjelloun. 3d structure and motion recovery in a multisensor framework. *Information Fusion*, 2(4):271 – 285, 2001. ISSN 1566-2535. doi: 10.1016/S1566-2535(01)00045-8.

[9] S. Agarwal, N. Snavely, S. Seitz, , and R. Szeliski. Bundle adjustment in the large. In *Computer Vision - ECCV 2010*, pages 29–42, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-15552-9.

[10] B. Lourenço, T. Madeira, P. Dias, V. Santos, and M. Oliveira. 2d lidar to kinematic chain calibration using planar features of indoor scenes. *Industrial Robot: the Int. journal of robotics research and application*, 47(5):647–655, Jan 2020. ISSN 0143-991X. doi: 10.1108/IR-09-2019-0201.

[11] A. Huang, M. Antone, E. Olson, L. Fletcher, D. Moore, S. Teller, and J. Leonard. A high-rate, heterogeneous data set from the darpa urban challenge. *The Int. Journal of Robotics Research*, 29(13):1595–1601, 2010. doi: 10.1177/0278364910384295.

[12] L. Huang and M. Barth. A novel multi-planar LIDAR and computer vision calibration procedure using 2d patterns for automated navigation. In *2009 IEEE Intelligent Vehicles Symposium*. IEEE, June 2009. doi: 10.1109/ivs.2009.5164263.

[13] Y. Liao, G. Li, Z. Ju, H. Liu, and D. Jiang. Joint kinect and multiple external cameras simultaneous calibration. In *2017 2nd Int. Conf. on Advanced Robotics and Mechatronics (ICARM)*, pages 305–310, Aug 2017. doi: 10.1109/ICARM.2017.8273179.

[14] J. Rehder, R. Siegwart, and P. Furgale. A general approach to spatiotemporal calibration in multisensor systems. *IEEE Trans. on Robotics*, 32(2):383–398, April 2016. ISSN 1552-3098. doi: 10.1109/TRO.2016.2529645.

[15] V. Pradeep, K. Konolige, and E. Berger. *Calibrating a Multi-arm Multi-sensor Robot: A Bundle Adjustment Approach*, pages 211–225. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. ISBN 978-3-642-28572-1. doi: 10.1007/978-3-642-28572-1_15.

[16] I. Ali, O. Suominen, A. Gotchev, and E. Morales. Methods for Simultaneous Robot-World-Hand–Eye Calibration: A Comparative Study. *Sensors*, 19(12), 2019. ISSN 1424-8220. doi: 10.3390/s19122837.

[17] M. Oliveira, A. Castro, T. Madeira, E. Pedrosa, P. Dias, and

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

V. Santos. A ROS framework for the extrinsic calibration of intelligent vehicles: A multi-sensor, multi-modal approach. *Robotics and Autonomous Systems*, page 103558, 2020. ISSN 0921-8890. doi: 10.1016/j.robot.2020.103558.

[18] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J Leibs, R. Wheeler, and A. Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009.

[19] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[20] C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for rgb-d cameras. In *Proc. of the Int. Conf. on Intelligent Robot Systems (IROS)*, 2013.

[21] P. Fankhauser, M. Bloesch, D. Rodriguez, R. Kaestner, M. Hutter, and R. Siegwart. Kinect v2 for mobile robot navigation: Evaluation and modeling. In *IEEE Int. Conf. on Advanced Robotics (ICAR)*, 2015.

[22] E. Marchand, F. Spindler, and F. Chaumette. Visp for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Robotics and Automation Magazine*, 12(4): 40–52, December 2005.

[23] S. Trinh, F. Spindler, E. Marchand, and F. Chaumette. A modular framework for model-based visual tracking using edge, texture and depth features. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS'18*, Madrid, Spain, October 2018.

[24] P. Furgale, J. Rehder, and R. Siegwart. Unified temporal and spatial calibration for multi-sensor systems. In *2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1280–1286, 2013.

[25] R. Tsai and R. Lenz. A New Technique for Fully Autonomous and Efficient 3D Robotics Hand/Eye Calibration. *IEEE Trans. on Robotics and Automation*, 5(3):345–358, jun 1989. ISSN 1042296X. doi: 10.1109/70.34770.

[26] R. Horaud and F. Dornaika. Hand-eye calibration. *The Int. Journal of Robotics Research*, 14(3):195–210, Jun 1995.

[27] R. Horaud and F. Dornaika. Hand-eye calibration. *The Int. Journal of Robotics Research*, 14(3):195–210, 1995. doi: 10.1177/027836499501400301.

[28] F. Dornaika and R. Horaud. Simultaneous robot-world and hand-eye calibration. *IEEE Trans. on Robotics and Automation*, 14(4):617–622, Aug 1998. ISSN 2374-958X. doi: 10.1109/70.704233.

[29] F. Park and B. Martin. Robot sensor calibration: solving AX=XB on the Euclidean group. *IEEE Trans. on Robotics and Automation*, 10(5):717–721, 1994. doi: 10.1109/70.326576.

[30] A. Liz, L. Wang, and D. Wu. Simultaneous robot-world and hand-eye calibration using dual-quaternions and Kronecker product. *Int. Journal of the Physical Sciences*, 5(10):1530–1536, 2010.

[31] M. Shah. Solving the Robot-World/Hand-Eye Calibration Problem Using the Kronecker Product. *Journal of Mechanisms and Robotics*, 5(3), 06 2013. ISSN 1942-4302. doi: 10.1115/1.4024473.

[32] A. Tabb and K. Yousef. Solving the robot-world hand-eye(s) calibration problem with iterative methods. *Machine Vision and Applications*, 28(5-6):569–590, aug 2017. ISSN 0932-8092. doi: 10.1007/s00138-017-0841-7.

[33] A. Malti. Hand–eye calibration with epipolar constraints: Application to endoscopy. *Robotics and Autonomous Systems*, 61 (2):161–169, February 2013. ISSN 0921-8890. doi: 10.1016/j.robot.2012.09.029.

[34] M. Oliveira, A. Castro, T. Madeira, P. Dias, and V. Santos. A General Approach to the Extrinsic Calibration of Intelligent Vehicles Using ROS. In *Robot 2019: Fourth Iberian Robotics Conf.*, pages 203–215, Cham, 2020. Springer Int. Publishing. ISBN 978-3-030-35990-4.

[35] Q. Zhang and 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)*, volume 3, pages 2301–2306. IEEE, 2004.

[36] Y. Zhang, G. Li, X. Xie, and Z. Wang. A new algorithm for accurate and automatic chessboard corner detection. In *2017 IEEE Int. Symposium on Circuits and Systems (ISCAS)*, pages 1–4, 2017. doi: 10.1109/ISCAS.2017.8050637.

[37] S. Jurado, R. Salinas, F. Cuevas, and R. Carnicer. Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern Recognition*, 51:481 – 491, 2016. ISSN 0031-3203. doi: 10.1016/j.patcog.2015.09.023.

[38] F. Ramirez, F. Salinas, and R. Carnicer. Speeded up detection of squared fiducial markers. *Image and Vision Computing*, 76:38 – 47, 2018. ISSN 0262-8856. doi: 10.1016/j.imavis.2018.05.004.

[39] D. Hu, D. DeTone, and T. Malisiewicz. Deep charuco: Dark charuco marker pose estimation. In *2019 IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 8428–8436, 2019.

[40] M. Ruan and D. Huber. Calibration of 3d sensors using a spherical target. In *2014 2nd Int. Conf. on 3D Vision*, volume 1, pages 187–193, 2014. doi: 10.1109/3DV.2014.100.

[41] M. Pereira, D. Silva, V. Santos, and P. Dias. Self calibration of multiple lidars and cameras on autonomous vehicles. *Robotics and Autonomous Systems*, 83:326 – 337, Sep 2016.

[42] D. Rato and V. Santos. Automatic registration of ir and rgb cameras using a target detected with deep learning. In *2020 IEEE Int. Conf. on Autonomous Robot Systems and Competitions (ICARSC)*, pages 287–293, 2020.

[43] Y. Kwon, J. Jang, and O. Choi. Automatic sphere detection for extrinsic calibration of multiple rgbd cameras. In *2018 18th Int. Conf. on Control, Automation and Systems (ICCAS)*, pages 1451–1454, Oct 2018.

[44] M. Almeida, P. Dias, M. Oliveira, and V. Santos. 3d-2d laser range finder calibration using a conic based geometry shape. In *Image Analysis and Recognition*, pages 312–319, Jun 2012.

[45] D. Gao, J. Duan, X. Yang, and B. Zheng. A method of spatial calibration for camera and radar. In *2010 8th World Congress on Intelligent Control and Automation*, pages 6211–6215, July 2010. doi: 10.1109/WCICA.2010.5554411.

[46] C. Guindel, J Beltrán, D. Martín, and F. García. Automatic extrinsic calibration for lidar-stereo vehicle sensor setups. *2017 IEEE 20th Int. Conf. on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2017.

[47] M. de Paula, C. Jung, and L. Silveira. Automatic on-the-fly extrinsic camera calibration of onboard vehicular cameras. *Expert Systems with Applications*, 41(4):1997–2007, March 2014. doi: 10.1016/j.eswa.2013.08.096.

[48] S. Álvarez, D. Llorca, and M. Sotelo. Hierarchical camera autocalibration for traffic surveillance systems. *Expert Systems with Applications*, 41(4):1532–1542, March 2014. doi: 10.1016/j.eswa.2013.08.050.

[49] G. Mueller and H. Wuensche. Continuous stereo camera calibration in urban scenarios. In *2017 IEEE 20th Int. Conf. on Intelligent Transportation Systems (ITSC)*, pages 1–6, Oct 2017. doi: 10.1109/ITSC.2017.8317675.

[50] L. Wu and B. Zhu. Binocular stereovision camera calibration. In *2015 IEEE Int. Conf. on Mechatronics and Automation (ICMA)*, pages 2638–2642, Aug 2015. doi: 10.1109/ICMA.2015.7237903.

[51] R. Su, J. Zhong, Q. Li, S. Qi, H. Zhang, and T. Wang. An automatic calibration system for binocular stereo imaging. In *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conf. (IMCEC)*,

pages 896–900, Oct 2016. doi: 10.1109/IMCEC.2016.7867340.

[52] Y. Ling and S. Shen. High-precision online markerless stereo extrinsic calibration. In *2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1771–1778, Oct 2016. doi: 10.1109/IROS.2016.7759283.

[53] V. Dinh, T. Nguyen, and J. Jeon. Rectification using different types of cameras attached to a vehicle. *IEEE Trans. on Image Processing*, 28(2):815–826, Feb 2019. ISSN 1057-7149. doi: 10.1109/TIP.2018.2870930.

[54] A. Khan, G. Camarasa, L. Sun, and 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)*, pages 1960–1965, Dec 2016. doi: 10.1109/ROBIO.2016.7866616.

[55] F. Basso, E. Menegatti, and A. Pretto. Robust intrinsic and extrinsic calibration of rgb-d cameras. *IEEE Trans. on Robotics*, 34(5):1315–1332, Oct 2018. ISSN 1552-3098. doi: 10.1109/TRO.2018.2853742.

[56] Y. Qiao, B. Tang, Y. Wang, and L. Peng. A new approach to self-calibration of hand-eye vision systems. In *2013 Int. Conf. on Computational Problem-Solving (ICCP)*, pages 253–256, Oct 2013. doi: 10.1109/ICCPS.2013.6893596.

[57] C. Zhang and Z. Zhang. Calibration between depth and color sensors for commodity depth cameras. In *2011 IEEE Int. Conf. on Multimedia and Expo*, pages 1–6, July 2011. doi: 10.1109/ICME.2011.6012191.

[58] G. Chen, G. Cui, Z. Jin, F. Wu, and X. Chen. Accurate intrinsic and extrinsic calibration of rgb-d cameras with gp-based depth correction. *IEEE Sensors Journal*, 19(7):2685–2694, April 2019. ISSN 1530-437X. doi: 10.1109/JSEN.2018.2889805.

[59] F. Vasconcelos, J. Barreto, and U. Nunes. A minimal solution for the extrinsic calibration of a camera and a laser-rangefinder. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 34 (11):2097–2107, Nov 2012.

[60] Q. Zhang and 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)*, volume 3, pages 2301–2306 vol.3, Sep. 2004. doi: 10.1109/IROS.2004.1389752.

[61] M. Häselich, R. Bing, and D. Paulus. Calibration of multiple cameras to a 3d laser range finder. In *2012 IEEE Int. Conf. on Emerging Signal Processing Applications*, pages 25–28, Jan 2012.

[62] Z. Chen, X. Yang, C. Zhang, and 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)*, pages 448–453, Oct 2016.

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

[64] G. Lee, J. Lee, and 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)*, pages 64–69, Nov 2017. doi: 10.1109/MFI.2017.8170408.

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

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

[67] W. Wang, K. Sakurada, and N. Kawaguchi. Reflectance intensity assisted automatic and accurate extrinsic calibration of 3d LiDAR and panoramic camera using a printed chessboard. *Remote Sensing*, 9(8):851, August 2017. doi: 10.3390/rs9080851.

[68] V. Fremont and P. Bonnifait. Extrinsic calibration between a multi-layer lidar and a camera. In *2008 IEEE Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems*. IEEE, August 2008. doi: 10.1109/mfi.2008.4648067.

[69] F. M. Mirzaei, D. G. Kottas, and S. I. Roumeliotis. 3d LIDAR–camera intrinsic and extrinsic calibration: Identifiability and analytical least-squares-based initialization. *The Int. Journal of Robotics Research*, 31(4):452–467, April 2012. doi: 10.1177/0278364911435689.

[70] G. Pandey, J. McBride, S. Savarese, and R. Eustice. Extrinsic calibration of a 3d laser scanner and an omnidirectional camera. *IFAC Proceedings Volumes*, 43(16):336–341, 2010. doi: 10.3182/20100906-3-it-2019.00059.

[71] J. Huang and J. Grizzle. Improvements to target-based 3d LiDAR to camera calibration. *IEEE Access*, 8:134101–134110, 2020. doi: 10.1109/access.2020.3010734.

[72] H. Zhuang, Z. Roth, and R. Sudhakar. Simultaneous robot/world and tool/flange calibration by solving homogeneous transformation equations of the form AX=YB. *IEEE Trans. on Robotics and Automation*, 10(4):549–554, Aug 1994. ISSN 2374-958X. doi: 10.1109/70.313105.

[73] L. Zhou, Z. Li, and M. Kaess. Automatic extrinsic calibration of a camera and a 3d LiDAR using line and plane correspondences. In *2018 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, October 2018. doi: 10.1109/iros.2018.8593660.

[74] A. Dhall, K. Chelani, V. Radhakrishnan, and K. M. Krishna. Lidar-camera calibration using 3d-3d point correspondences, 2017.

[75] E. Kim and S. Park. Extrinsic calibration between camera and LiDAR sensors by matching multiple 3d planes. *Sensors*, 20(1): 52, December 2019. doi: 10.3390/s20010052.

[76] H. Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics & Automation Magazine*, 13(2): 99–110, June 2006. doi: 10.1109/mra.2006.1638022.

[77] V. Santos, J. Almeida, E. Ávila, D. Gameiro, M. Oliveira, R. Pascoal, R. Sabino, and 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*, pages 1421–1427, Sep. 2010. doi: 10.1109/ITSC.2010.5625031.

[78] V. Santos, D. Rato, P. Dias, and M. Oliveira. Multi-sensor extrinsic calibration using an extended set of pairwise geometric transformations. *Sensors*, 20(23), 2020. ISSN 1424-8220. doi: 10.3390/s20236717.

[79] D. Noel, J. Sarmiento, R. Ojeda, and J. Jimenez. Automatic multi-sensor extrinsic calibration for mobile robots. *IEEE Robotics and Automation Letters*, 4(3):2862–2869, July 2019. doi: 10.1109/lra.2019.2922618.

[80] A. Aguiar, M. Oliveira, E. Pedrosa, and F. Santos. A camera to lidar calibration approach through the optimization of atomic transformations. *Expert Systems with Applications*, page 114894, 2021. ISSN 0957-4174. doi: 10.1016/j.eswa.2021.114894. IF 5.423 (2020).

[81] E. Pedrosa, M. Oliveira, N. Lau, and V. Santos. A general approach to hand–eye calibration through the optimization of atomic transformations. *IEEE Trans. on Robotics*, pages 1–15, 2021. doi: 10.1109/TRO.2021.3062306. IF 6.123 (2020).

[82] J. Hornegger and C. Tomasi. Representation issues in the ml estimation of camera motion. In *Proc. of the IEEE Int. Conf. on Computer Vision*, volume 1, pages 640 – 647 vol.1, 02 1999. ISBN 0-7695-0164-8. doi: 10.1109/ICCV.1999.791285.

[83] M. Branch, T. Coleman, and Y. Li. A subspace, interior, and conjugate gradient method for Large-Scale Bound-

Constrained minimization problems. *SIAM J. Sci. Comput.*, 21(1):1–23, January 1999. ISSN 1064-8275. doi: 10.1137/S1064827595289108.

[84] T. Foote. tf: The transform library. In *2013 IEEE Conf. on Technologies for Practical Robot Applications (TePRA)*, pages 1–6, April 2013. doi: 10.1109/TePRA.2013.6556373.

[85] X. Gao, X. Hou, J. Tang, and H. Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25:930–943, 2003.

[86] A. Sanchez, J. Cetto, and F. Noguer. Exhaustive linearization for robust camera pose and focal length estimation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 35(10):2387–2400, 2013.