An Extrinsic Sensor Calibration Framework for Sensor-fusion based Autonomous Vehicle Perception

Mokhtar Bouain[1,2], Denis Berdjag[2], Nizar Fakhfakh[1] and Rabie Ben Atitallah[2]
[1]*Navya Company, Paris, France*
[2]*LAMIH CNRS UMR 8201, University of Valenciennes, 59313 Valenciennes, France*

Keywords: Sensor Alignment, Sensor Calibration, Sensor Fusion, Intelligent Vehicles.

Abstract: In this paper we deal with sensor alignment problems that appear when implementing sensor fusion-based autonomous vehicle perception. We focus on extrinsic calibration of vision-based and line scan LIDAR sensors. Based on state-of-art solutions, a consistent calibration toolchain is developed, with improvements (accuracy and calibration duration). Additionally, sensor alignment/calibration impact on fusion-based perception is investigated. Experimental results are provided for illustration, using real-world data.

1 INTRODUCTION

When dealing with robotic perception, single-sensor architecture are range-limited. This limit depends on many factors such as the technology of the device (resolution, range...), limited spatial and temporal coverage, measurement rates, noises... Indeed, any cost-efficient sensor will be optimized to deal with a specific task. As a result, when dealing with perception tasks in a rich environment, such as object detection and avoidance for robots or autonomous navigation, using multiple sensors is a natural solution (Baig et al., 2011). Multi-sensor perception requires data fusion approaches to reconstruct world features for the robot, based on synergistic and redundant measurements. Data fusion is indeed widely used in many fields of robotics such as perception (obstacle detections, environment mapping)(Wittmann et al., 2014) and also in process control tasks. However combining homogeneous or heterogeneous measurements remains a challenge to be tackled. Some of the key issues are the diversity of the existing technologies and the appearance of new sensor types. Other issues are related to the application field, for example, autonomous vehicles will be driven in ”open” environments, and that implies stringent security constraints. This research deals with the initial phase of any multi-sensor acquisition, the alignment process. When features are acquired for real world measurements, normalization is required in order to reconstruct the IA perceived word without bias, and take appropriate actions. We address specifically vision-based and LIDAR based sensor alignment, and derive a general framework and the appropriate toolchain. Despite the popularity topic, discussed in the next section, few works address all the steps of the process. Then we discuss the impacts of the calibration procedure on fusion accuracy. This paper present three main contributions:

- We develop further a specific framework presented in (Guo and Roumeliotis, 2013), and extend it from single to multi-line reference in order to reduce the required number poses. In addition we address the problem of point-normal vector correspondences.
- we present a complete implementable toolchain, to extract the co-features for both types of sensors: line detections for cameras and segmentation process for LIDAR sensors in order to make fully automated feature acquisition.
- we investigate the impact of the calibration accuracy on sensor fusion performance.

The remaining of this paper is organized as follows: In section 2 we present a survey of existing methods and point out the classification of alignment methods. In section 3 we describe the problem formulation and we present the analytical least square solution to the multi-line calibration approach. The co-feature extraction is discussed in section 4. We detail the impact of the calibration task of sensor-fusion accuracy in section 5. Section 6 discusses the experimental setup and tests using real data. The conclusion and future work are discussed in section 7.

2 RELATED WORK

Multi-sensor architectures are popular nowadays, and research works on derived topics such as calibration are also plentiful, especially about visual, inertial and LIDAR sensors. The authors of (Li et al., 2013) classify extrinsic calibration methods of camera and LIDAR sensors. According to this classification, there are three categories of camera/LIDAR calibration: the first method is based on auxiliary sensors; using a third sensor which is Inertial Measurement Unit (IMU), extrinsic calibration is carried out. It is shown that the rigid transformation between the two frames can be estimated using the IMU (Nez et al., 2009). The second method is based on specially designed calibration boards. The idea is to use a particular pattern to determine targets position in both sensors frames, and subsequently express the target coordinates in each frame in order to derive the rigid transformations. In (Fremont et al., 2012), the calibration uses circular targets for intelligent vehicle applications and dedicated to multi-layer LIDAR. This method determines the relative pose in rotation and translation of the sensors using sets of corresponding circular features acquired for several target configurations (Fremont et al., 2012). Similarly, (Park et al., 2014) uses a polygonal planar board to perform calibration of color camera and multi-layer Lidar for robot navigation tasks. Concerning the third category, it is about methods that use chessboard targets. This kind of calibration is also pattern specific. The advantage of this method is determining the intrinsic parameters simultaneously for cameras and extrinsic calibration of the camera and the LIDAR (Li et al., 2013). In addition to these three categories, we consider another category which is the automatic extrinsic calibration. This kind of method is handled without a designed calibration board or another sensor as mentioned above. (John et al., 2015) proposes a calibration approach does not need any particular shape to be located. Their method consists to integrate the perceived data from 3D LIDAR and stereo camera using Particle Swarm Optimization algorithms (PSO), using acquired objects from the outer world, without the aid of any dedicated external pattern.

In addition, we can distinguish three main methods to solve the established closed form between the correspondence features. The first one solves the closed form using linear methods such as the Singular Value Decomposition (SVD) and uses this solution as a first guess to perform a nonlinear optimization such as Gauss-Newton or Levenberg-Marquardt algorithms. The second method is based on the idea that the determination of the global minimum of a given cost function needs to find an initial guess located in the basin of attraction. The authors of (Guo and Roumeliotis, 2013) proposed an analytical least-squares approach to carry out a generic calibration process. The third method uses stochastic approaches or search algorithms to associate the features between two frames as the PSO algorithm, as it is shown in (John et al., 2015). Based on the literature review, we believe that the development of a generic solution for sensor alignment is a viable solution. However in the literature, little is said on the relationship between sensor calibration and sensor fusion steps, apart from automatic calibration approaches. We believe that such contribution, for target-based solutions (shape or pattern specific), is useful, especially if computation-heavy algorithms (such as PSO) are avoided. In addition, few works address the tool-chain implementation on real vehicles. We believe that this topic is of interest for practitioners.

3 PROBLEM FORMULATION AND ANALYTICAL LEAST-SQUARES SOLUTION

3.1 Problem Formulation and Basic Concepts

The calibration process is an alignment procedure of a given sensor frames. That is to say, find the relation between the coordinates of sensor frames to ensure the transformation from a frame into another. Concerning the extrinsic calibration of a LIDAR sensor and camera, it is about estimation of the relative position for a given point located in the real world frame, in the LIDAR and camera frames. the objective is to find the unknown 6 Degrees Of Freedom (DOF) transformation between the two sensor frames. In other words, the goal is to find the rigid transformation $[{}^{C}R_{L}|{}^{C}\overrightarrow{t}_{L}]$, which allows us to determine the correspondence of a given 3D LIDAR point represented as $\overrightarrow{p}_{L} = [x_L, y_L, z_L]^T$ located into the frame of the LIDAR sensor $\{L\}$, in the frame of the camera $\{C\}$. Let $\overrightarrow{p}_{C} = [x_C, y_C, z_C]^T$ be the correspondence of $\overrightarrow{p}_{L}$:

$$\overrightarrow{p}_{C} = {}^{C}R_{L}\overrightarrow{p}_{L} + {}^{C}\overrightarrow{t}_{L} \quad (1)$$

Based on (Guo and Roumeliotis, 2013), we extend the existing calibration solution to multi-line pattern targets. We define the coordinate system for the sensors as follows: the origin O_C is the center of the camera and the origin O_L is the center of the LIDAR sensor frame. Without loss of generality, the LIDAR scanning plane is defined as the plane $z_L = 0$ (see figure 1). Thus, a 3D LIDAR point represented as $\overrightarrow{p}_L = [x_L, y_L, 0]^T$. We consider that the calibration board contains d horizontal black lines lb_{ih} where h is the number of line and i is the number of pose. It is required to fix these lines equidistant between each other, i.e. if there are d lines, it is necessary that the $(d+1)$ parts are equals. Assume that $\overrightarrow{p}_{li}$ and $\overrightarrow{p}_{ri}$ are respectively the left and right ending points of the pattern. By dividing the distance between $\overrightarrow{p}_{li}$ and $\overrightarrow{p}_{ri}$ by $(d+1)$, we get an estimation of the positions of $\overrightarrow{p}_{mih}$ (Fig. 1). Otherwise, the normal vector $\overrightarrow{n}_{ih}$ is perpendicular to the plane T_h defined by l_{bih} and the camera center. Since the correspondent of $\overrightarrow{p}_{mih}$ in the camera frame $\{C\}$ belongs to the plane T_h, then $\overrightarrow{n}_{ih}$ will correspond to $\overrightarrow{p}_{mih}$. Therefore, we obtain the following geometric constraints:

$$\overrightarrow{n}_{ih}^T \overrightarrow{p}_{cih} = \overrightarrow{n}_{ih}^T ({}^C R_L \overrightarrow{p}_{mih} + {}^C\overrightarrow{t}_L) = 0 \qquad (2)$$

where $\overrightarrow{p}_{cih}$ is the correspondent of the LIDAR point $\overrightarrow{p}_{mih}$ in the camera frame.

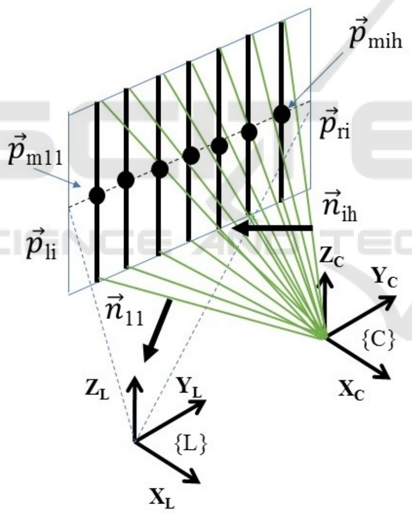

Figure 1: Notations and geometric constraints.

In real-world situations (2) is never satisfied due to the measurement noise and it will usually be slightly different from zero. The difference is the error e_i. All the residuals for n measurements can be represented as

$$\overrightarrow{e} = (\overrightarrow{n}_{11}^T ({}^C R_L \overrightarrow{p}_{m11} + {}^C\overrightarrow{t}_L), \overrightarrow{n}_{12}^T ({}^C R_L \overrightarrow{p}_{m12} + {}^C\overrightarrow{t}_L), \ldots \overrightarrow{n}_{nd}^T ({}^C R_L \overrightarrow{p}_{mnd} + {}^C\overrightarrow{t}_L)) \qquad (3)$$

To estimate the transformation parameters $[{}^C R_L | {}^C\overrightarrow{t}_L]$ with accurately and minimize the residuals from (3), we define the cost function J and we aim to minimize the sum of squared errors :

$$J = \underset{{}^C t_L, {}^C R_L}{argmin} \sum_{i=1}^{n} \sum_{h=1}^{d} (e_{ih})^2 \qquad (4)$$

$$J = \underset{{}^C t_L, {}^C R_L}{argmin} \sum_{i=1}^{n} \sum_{h=1}^{d} (\overrightarrow{n}_{ih}^T ({}^C R_L \overrightarrow{p}_{mih} + {}^C\overrightarrow{t}_L))^2$$
$$s.t. {}^C R_L^T {}^C R_L = I, \ \det({}^C R_L) = 1 \qquad (5)$$

The two above conditions represent the rotational matrix constraints.
We now have a correspondence between a 3D point in LIDAR frame and the plane defined by O_C and the black lines in the camera frame. First, according to (2) and for $h = 1...d$, ${}^C R_L$ and ${}^C\overrightarrow{t}_L$ are the unknowns. Second, let $\overrightarrow{r}_1$, $\overrightarrow{r}_2$ and $\overrightarrow{r}_3$ be the three columns of ${}^C R_L$. Since the LIDAR scanning plane is defined as the plane $z_L = 0$ then we do not have an explicit dependence on $\overrightarrow{r}_3$ and hence we can rewrite $\overrightarrow{r}_3 = \overrightarrow{r}_1 \times \overrightarrow{r}_2$ ($\times$ is the cross product).

To summarize, we get nine unknowns grouped in $\overrightarrow{r}_1$, $\overrightarrow{r}_2$ and ${}^C\overrightarrow{t}_L$ and three constraints since ${}^C R_L$ is an orthonormal matrix.

3.2 Closed-form Solution

3.2.1 Redefining the Optimization Problem

The objective is to solve the equation (5) to find the parameters of calibration. The first step consists to reduce the number of variables and constraints in (5). Since the translation, ${}^C\overrightarrow{t}_L$ is not involved in the constraints, it can be eliminated from the optimization problem.
Lemma 1: By reducing the number of variables in (5) the cost function is defined as follows:

$$J = \sum_{i=1}^{n} \sum_{h=1}^{d} \left[\overrightarrow{n}_{ih}^T {}^C R_L \overrightarrow{p}_{mih} - (\sum_{j=1}^{n} \sum_{l=1}^{d} \overrightarrow{w}_{ihjl}^T {}^C R_L \overrightarrow{p}_{mjl}) \right]^2$$
$$s.t. {}^L R_C^T {}^C R_L = I, \ \det({}^C R_L) = 1 \qquad (6)$$

where:

$$\overrightarrow{w}_{ihjl} = \overrightarrow{n}_{ih}^T \Big(\sum_{j=1}^{n} \sum_{l=1}^{d} \overrightarrow{n}_{jl} \overrightarrow{n}_{jl}^T \Big)^{-1} \overrightarrow{n}_{jl} \overrightarrow{n}_{jl}^T$$

$${}^C\overrightarrow{t}_L = -\Big(\sum_{i=1}^{n} \sum_{h=1}^{d} \overrightarrow{n}_{ih} \overrightarrow{n}_{ih}^T \Big)^{-1} \Big(\sum_{i=1}^{n} \sum_{h=1}^{d} \overrightarrow{n}_{ih} \overrightarrow{n}_{ih}^T {}^C R_L \overrightarrow{p}_{mih} \Big) \qquad (7)$$

Proof: By applying the first order necessary condition for optimality to the cost function, we obtain:

$$\frac{\partial J}{\partial {}^C\overrightarrow{t}_L} = \frac{\partial}{\partial {}^C\overrightarrow{t}_L} \Big(\sum_{i=1}^{n} \sum_{h=1}^{d} (\overrightarrow{n}_{ih}^T ({}^C R_L \overrightarrow{p}_{mih} + {}^C\overrightarrow{t}_L))^2 \Big)$$
$$= \sum_{i=1}^{n} \sum_{h=1}^{d} 2 \overrightarrow{n}_{ih} \left[\overrightarrow{n}_{ih}^T {}^C R_L \overrightarrow{p}_{mih} + \overrightarrow{n}_i^T {}^C\overrightarrow{t}_L \right] = 0$$

$${}^{C}\overrightarrow{t}_L = -F^{-1}\Big(\sum_{i=1}^{n}\sum_{h=1}^{d}\overrightarrow{n}_{ih}\overrightarrow{n}_{ih}^{T}\,{}^{C}R_L\overrightarrow{p}_{mih}\Big) \qquad (8)$$

Where :

$$F = \Big(\sum_{i=1}^{n}\sum_{h=1}^{d}\overrightarrow{n}_{ih}\overrightarrow{n}_{ih}^{T}\Big)$$

$$J = \sum_{i=1}^{n}\sum_{h=1}^{d}\Big[\overrightarrow{n}_{ih}^{T}\,{}^{C}R_L\overrightarrow{p}_{mih} - \overrightarrow{n}_{ih}^{T}F^{-1}\big(\sum_{j=1}^{n}\sum_{l=1}^{d}\overrightarrow{n}_{jl}\overrightarrow{n}_{jl}^{T}\,{}^{C}R_L\overrightarrow{p}_{mjl}\big)\Big]^2$$
$$= \sum_{i=1}^{n}\sum_{h=1}^{d}\Big[\overrightarrow{n}_{ih}^{T}\,{}^{C}R_L\overrightarrow{p}_{mih} - \big(\sum_{j=1}^{n}\sum_{l=1}^{d}\overrightarrow{n}_{ih}^{T}F^{-1}\overrightarrow{n}_{jl}\overrightarrow{n}_{jl}^{T}\,{}^{C}R_L\overrightarrow{p}_{mjl}\big)\Big]^2$$

And hence, the cost function is expressed only by the rotation matrix and its constraints:

$$J = \sum_{i=1}^{n}\sum_{h=1}^{d}\Big[\overrightarrow{n}_{ih}^{T}\,{}^{C}R_L\overrightarrow{p}_{mih} - \big(\sum_{j=1}^{n}\sum_{l=1}^{d}\overrightarrow{w}_{ihjl}^{T}\,{}^{C}R_L\overrightarrow{p}_{mjl}\big)\Big]^2$$

3.2.2 Simplifying Optimization Problem using Quaternion Unit

To simplify the problem and reduce further the number of unknowns, the quaternion unit q is employed to represent the rotation matrix ${}^{C}R_L$. The conversion from vectors to quaternions and all the quaternion parameterization are presented in the appendix.

Lemma 2: By using the quaternion units, (6) can be written as follows:

$$J = \sum_{i=1}^{n}\sum_{h=1}^{d}\Big[q^T S_{ih}\, q\Big]^2 \qquad (9)$$
$$s.t.\ {}^{L}q_C^{T}\,{}^{L}q_C = 1$$

where :

$$S_{ih} = \mathcal{L}(\bar{n}_{ih})^T\mathcal{R}(\bar{p}_{mih}) - \sum_{j=1}^{n}\sum_{l=1}^{d}\mathcal{L}(\bar{w}_{ihjl})^T\mathcal{R}(\bar{p}_{mjl})$$

where $\mathcal{L}(.)$ and $\mathcal{R}(.)$ are left and right quaternion multiplication matrices (see appendix for details).

Proof: Based on quaternion calculus properties, conversion of 3D vectors into quaternions and quaternion multiplication, we can rewrite the cost function as follows:

$$J = \sum_{i=1}^{n}\sum_{h=1}^{d}\Big[\bar{n}_{ih}^{T}(q\otimes\bar{p}_{mih}\otimes q^{-1}) - \sum_{j=1}^{n}\sum_{l=1}^{d}\bar{w}_{ihjl}^{T}(q\otimes\bar{p}_{mjl}\otimes q^{-1})\Big]^2$$
$$= \sum_{i=1}^{n}\sum_{h=1}^{d}\Big[\bar{n}_{ih}^{T}(\mathcal{L}(q)\bar{p}_{mih}\otimes q^{-1}) - \sum_{j=1}^{n}\sum_{l=1}^{d}\bar{w}_{ihjl}^{T}(\mathcal{L}(q)\bar{p}_{mjl}\otimes q^{-1})\Big]^2$$

Since $(\mathcal{L}(q)\bar{p}_{mih})\otimes q^{-1} = \mathcal{R}(q^{-1})(\mathcal{L}(q)\bar{p}_{mih})$

and $(\mathcal{L}(q)\bar{p}_{mjl})\otimes q^{-1} = \mathcal{R}(q^{-1})(\mathcal{L}(q)\bar{p}_{mjl})$

Then :

$$J = \sum_{i=1}^{n}\sum_{h=1}^{d}\Big[\bar{n}_{ih}^{T}\mathcal{R}(q^{-1})\mathcal{L}(q)\bar{p}_{mih} - \sum_{j=1}^{n}\sum_{l=1}^{d}\bar{w}_{ihjl}^{T}\mathcal{R}(q^{-1})\mathcal{L}(q)\bar{p}_{mjl}\Big]^2$$

Since $\mathcal{L}(q)\,\bar{p}_{mih} = \mathcal{R}(\bar{p}_{mih})q$

and $\mathcal{L}(q)\,\bar{p}_{mjl} = \mathcal{R}(\bar{p}_{mjl})q$

Then the the cost function is reformulated as the following :

$$J = \sum_{i=1}^{n}\sum_{h=1}^{d}\Big[q^T\mathcal{L}(\bar{n}_{ih})^T\mathcal{R}(\bar{p}_{mih})\,q - \sum_{j=1}^{n}\sum_{l=1}^{d}q^T\mathcal{L}(\bar{w}_{ihjl})^T\mathcal{R}(\bar{p}_{mjl})\,q\Big]^2$$
$$= \sum_{i=1}^{n}\sum_{h=1}^{d}\Big[q^T\Big(\mathcal{L}(\bar{n}_{ih})^T\mathcal{R}(\bar{p}_{mih}) - \sum_{j=1}^{n}\sum_{l=1}^{d}\mathcal{L}(\bar{w}_{ihjl})^T\mathcal{R}(\bar{p}_{mjl})\Big)q\Big]^2$$

3.2.3 Lagrange Multiplier Method

In order to solve the equation form Lemma 2, we show now how to formulate the problem as a set of polynomial equations. To solve the cost function J we use the Lagrange multiplier method. According to the Lagrange method, the Lagrangian function is defined as follows:

$$L(q,\lambda) = J(q) + \lambda(\overset{T}{q}q - 1) \qquad (10)$$

Where λ is the Lagrange multiplier.
Hence, using the method of Lagrange multiplier, we obtain the following equations:

$$\begin{cases} \sum_{i=1}^{n}\sum_{h=1}^{d}\Big[q^T S_{ih}\,q\Big]\Big[S_{ih}+S_{ih}^{T}\Big]q + \lambda q = 0 \\ \overset{T}{q}q - 1 = 0 \end{cases} \qquad (11)$$

3.3 The Problem of Point-normal Vector Correspondences

The use of point-normal vector correspondences is strongly impacted by the quality of LIDAR endpoint detection. Sometimes, the LIDAR endpoint will not exactly locate on the border of the calibration target. In order to overcome this problem, the authors of (Lipu and Zhidong, 2014) have proved that placing the calibration target nearby the LIDAR sensor provides a high quality line-point correspondence and sufficient constraints to estimate the rigid transformation. In addition, we present a method to improve the accuracy of the endpoint estimation using virtual points. The positions of the virtual points are determined by the average distance between the points of LIDAR for each pose. So, if $\overrightarrow{p}_{li}$ and $\overrightarrow{p}_{ri}$ are the ending points of calibration board (left and right side), the average Euclidean distance is calculated as follows :

$$d_{lri} = \frac{dist(\overrightarrow{p}_{li},\overrightarrow{p}_{ri})}{n} \qquad (12)$$

where n is the number of scanned point located on the pattern. So, the positions of the left and right virtual points are :

$$\overrightarrow{p}_{Vli} = \overrightarrow{p}_{li} - \frac{d_{lri}}{2}\overrightarrow{u}$$
$$\overrightarrow{p}_{Vri} = \overrightarrow{p}_{ri} + \frac{d_{lri}}{2}\overrightarrow{u}$$

where $\overrightarrow{u} = \overrightarrow{p}_{ri} - \overrightarrow{p}_{li}$

4 PREPROCESSING AND FEATURE EXTRACTION

4.1 Extraction of 3D LIDAR Points

To extract the projected points of the source sensor on the calibration board, the automatic extraction approach by differentiation of the measurements and background data in static environments is often used. However, in this work, this task is carried out by using a segmentation process. Each segment (cluster) is defined as a set of points and is composed ofa minimum number of points distant according to a threshold distance denoted Thr. Therefore, if $dist(\overrightarrow{p}_i, \overrightarrow{p}_{i+1}) < Thr$ then a segment is defined with C_i as its centroid. Where, $\overrightarrow{p}_i$ is the impact point of the LIDAR sensor, $dist(\overrightarrow{p}_i, \overrightarrow{p}_{i+1})$ is the Euclidean distance between two adjacent points and Thr is the required threshold. The coordinate of each centroid C_i is calculated as follows: $(\sum \frac{p_{x_i}}{n}, \sum \frac{p_{y_i}}{n})$ where n is the number of points. We can add another parameter to fix the minimum number of points that compose a segment. Figure 2 shows the projected impact points on the target and its environment.

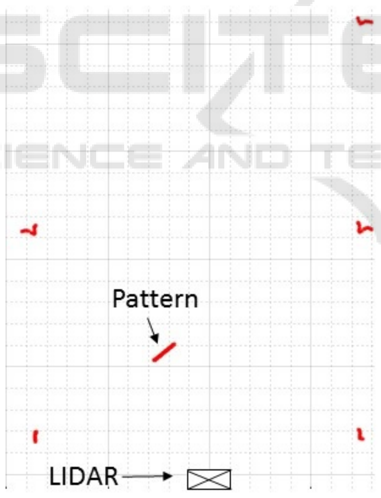

Figure 2: Projected impact points (environment).

4.2 Extraction of Lines using Hough Transform

Hough transform is considered to be an efficient method to locate lines. The idea is to transform every point in $x-y$ space (Cartesian frame) into parameter space (Polar frame). This method defines two parameters spaces which are r the length of a normal from the origin of this line and θ is the orientation of r with respect to the $x-axis$. Hence the line equation for each line is $r = x\cos(\theta) + y\sin(\theta)$. After the transformation of all points into the parameter space, local peaks in the parameter space associated to line candidates in $x-y$ space can be extracted. Otherwise, the line detection chain consists of a set of instructions. Before using the Hough transform it is necessary to apply a spatial filter to the image in order to reduce the noise. Among existing spatial filters, we use the Gaussian smoothing which is a 2-D filter of images that is used to remove/reduce the details and noises and also to blur images. Mathematically, applying the Gaussian smoothing filter it is the same as convolving the image with a Gaussian kernel. The main idea is that the new pixels of the image are created by a weighted average of the pixels close to it (gives more weight to the central pixels and less weights to the neighbors). After the removal of details and noises, the edges of the image will be extracted using an edge detector algorithm. It allows us to find the boundary of objects and hence extract useful structural information in order to reduce the amount of data to be processed. Mainly there are two commonly used approaches for edge detection which are Canny and Sobel edge detector (the difference of these approaches is the kernel in that they are using). In this work, we use the Canny edge detector. It is based on finding the intensity gradient of the image, and according to fixed thresholds, a pixel will be accepted as an edge or rejected. At this stage, the Hough transform is applied. Note that there are additional steps are taken to perform the extraction of black lines, such as limiting the regions of interest to reduce the computational burden. Also, it is necessary to filter false detections and keep only the inlier lines. To do this, we use a priori knowledge because we know that we need only the vertical lines and concerning the horizontal lines (or close to be), they are eliminated according to their slopes. In addition, we use another criterion which is the color of lines that are looking which is the black therefore we keep the lines that have the black color.

5 SENSOR CALIBRATION ACCURACY IMPACT ON MULTI-SENSOR DATA FUSION

In order to fuse the data between sensors, it is necessary to estimate or model the errors that are involved in the data processing level. These errors will be used to represent the uncertainties of sensors and to weigh the measurements during the fusion process. Sensor uncertainties are caused by many types of errors. We distinguish two main types: the random errors which are the noise measurements and the calibration errors which are caused by the alignment process. Therefore a multi-sensor data fusion should take into account these errors to improve its quality. The authors of (Baig et al., 2011) use the Bayesian Fusion technique to fuse the positions acquired by two sensors in the context of environment perception of autonomous vehicles.

The sensors are employed to detect the positions of obstacles. Position uncertainty is represented using 2D Gaussian distribution for both objects. Therefore, if X is the true position of the detected object, by using the Bayesian fusion, the probability of fused position $P_F[x_F\, y_F]^T$ by the two sensors is given as:

$$P_{rob}(P|X) = \frac{e^{\frac{-(P-X)^T R^{-1}(P-X)}{2}}}{2\pi\sqrt{|R|}} \tag{13}$$

where P is the fused position and R is the covariance matrix are given as :

$$P = \frac{P_1/R_1 + P_2/R_2}{1/R_1 + 1/R_2} \text{ and } 1/R = 1/R_1 + 1/R_2$$

where P_1 and R_1 are the position and covariance matrix of sensor 1 and P_2 and R_2 are that of sensor 2. We acquire positions of the detected obstacles by both sensors. Figure 3 shows the modeled error positions of one detected obstacle by two sensors. The cross represents the real positions which are unknown, the two black dots represent the measurements of positions, the circles (red and green) are the position uncertainties. The red circle is the calibration uncertainty which is generated by the calibration process which we are aiming to minimize. We test the impact of the calibration process on a multi-sensor data fusion based on Bayesian approach by varying the covariance matrices of sensors. The noise measurements will be Gaussian noises with zero mean and the standard deviation is 120 mm for both of sensors. The dynamic model of position is modeled by a rectilinear motion. Note that $R_1 = R_{m1}$ i.e. the covariance matrix of sensor 1 contains only the error of measurements. In contrast, $R_2 = R_{m2} + R_c$ i.e. the covariance of sensor 2 contains both measurement and calibration errors (because the measurements of sensor 2 will be projected on the frame of sensor 1). To summarize, in the first simulation, the calibration error of sensor 2 will be small while in the second simulation we will increase its calibration error.

Figures 4 and 5 show the obtained results: the norm of position (meter) for each sensor with two configurations of calibration errors over time. It is clear that in the first simulation (Figure 4: when the sensor 2 has a small value of calibration error), the fused position is located between the two provided positions by

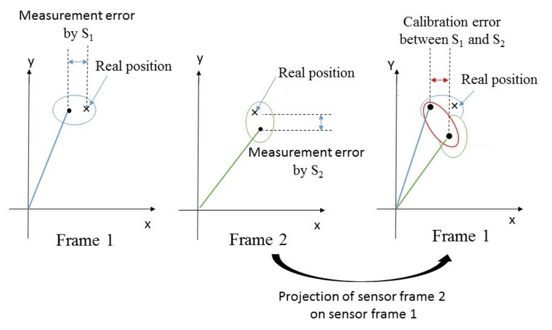
Measurement error by S$_1$
Real position
y
x
Frame 1
y
Real position
Measurement error by S$_2$
x
Frame 2
Calibration error between S$_1$ and S$_2$
Y
Real position
x
Frame 1
Projection of sensor frame 2 on sensor frame 1

Figure 3: Measurement and calibration errors.

the sensors. In contrast, in the second simulation and when the calibration error rises, the fused position follows the position provided by sensor 1, because it has the smallest error (see figure 5). According to these experiments, it is clear that the outcome is a combination of the two measurements weighted by their noise covariance's matrices. Therefore, if the calibration error grows then the projection of the measurement of sensor 2 will be distorted, so the combined result follows uniquely the measurement of the first sensor (Figure 5). This makes the use of sensor 2 obsolete and demonstrates the advantages of the multi-sensor data fusion architecture.

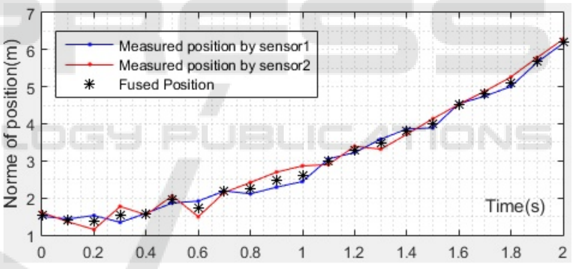

Figure 4: Fusion of two positions weighted with similar covariance matrices.

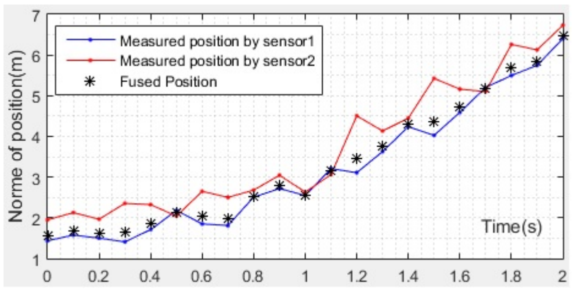

Figure 5: Fusion of two positions with the increase in the calibration error of sensor 2.

In practice, it is difficult to obtain or evaluate the ground truth of the real extrinsic parameters between the camera and LIDAR sensors. There are some works define theirs own criteria. For our work we propose the sum of squared residuals as an indicator of calibration. As long as this criterion tends to 0 we will get a good performance of calibration process.

6 EXPERIMENTS

In order to validate the multi-line approach, we conducted a series of experiments in the real environment. We use line-scan LIDAR with an angular resolution 0.25 degree and a color camera with 640x480 resolution. The camera is modeled using a pinhole model. The number of point-normal vector correspondences is fixed to three. We use a white calibration board with three black lines ($d = 3$). Note that a chessboard pattern is also usable.

6.1 Extraction of Lines from Calibration Board

Figure 6 shows the calibration board with three black lines. We use an image processing algorithm based on Hough Transform to extract the lines. Figure 7 shows a partial result that contain some false detections other than the black lines (Figure 8). To remove the false detections, we perform filtering and keep just the three black lines (see section 6.1). Therefore, from each line we need two points to determine the normal vectors.

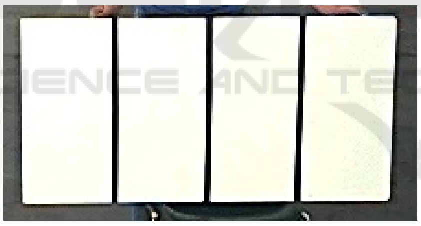
Figure 6: The used pattern for calibration (3 black lines).

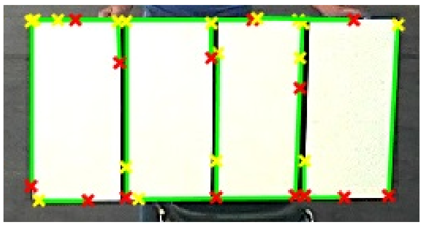
Figure 7: Line detections using Hough Transform.

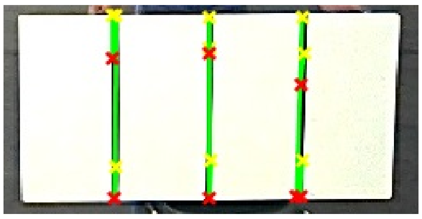
Figure 8: Keeping the black lines of the calibration board.

6.2 Results of Camera LIDAR Calibration

For the original (existing) approach, we collected 21 point-normal vector correspondences to estimate calibration parameters and used only 7 poses for the modified approach. The calibration board was moved to a 3m to 9m distance range. To compare the results, the average between the pixel coordinates of LIDAR points for different poses is calculated. Table 1 shows the results. The shown results correspond to the average absolute distance for u axis, v axis and the average Euclidean distance between the correspond pixels. According to these results, the two implementations are very close. Therefore, the result of the multi-lines approach is well performed as original approach with benefits reducing the required number of poses.

Otherwise, in order to suppress or at least reduce the effect of noise during measurements, it is reasonable to use multiple observations of the calibration pattern from different views to obtain the required 6 DOF. Also, it is required to rotate the calibration board to allow normal vector $\overrightarrow{n}_{ih}$ to span all three directions.

Figure (9) shows the projection results for the two approaches for different orientations. Visually, it is clear that there is a small difference between the projected points for both of methods. It is due to the different noises measurements and the polynomial equation solver behavior. In fact, equation (11) has a floating-point coefficients and it is not possible to get the same solution because the measurements and noises will be varying from one data set to another.

Table 1: The average difference between the two approaches.

Mean "u" dist. (pixel)	Mean "v" dist. (pixel)	Mean dist. (pixel)
2.399	3.6816	3.3927

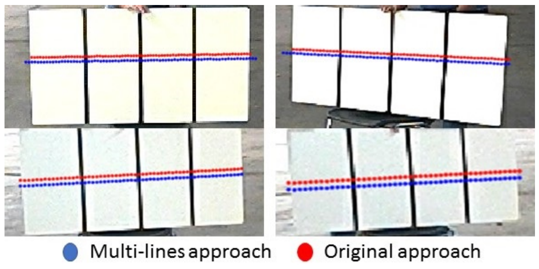

Figure 9: Projection of the LIDAR points into the image plane.

7 CONCLUSIONS

In this paper, we address the problem of the frame alignment between a camera and a 2-D LIDAR sensor through the extended generic framework. The original problem formulation implies using a given number of calibration poses is improved to use less poses. When compared to other approaches which accuracy depends on a precise initial guess, the proposed solution is formal and gives an optimal result for a given calibration poses batch. We proved that with such method, the number of observations can be reduced for an accurate result. In addition, a tool-chain is derived to extract sensor-acquired co-features. Real experiments confirmed that the presented development is beneficial for our application. Future works will deal with a native integration of the alignment process in the fusion algorithm, with an expected real-time sensor realignment module to deal with inaccurate initial calibration.

ACKNOWLEDGEMENTS

The authors would like to think Navya company who provided the data set and the platform of experiments. This research is financially supported by Navya company and ANRT CIFRE program grant from the French Government.

REFERENCES

Baig, Q., O., A., Trung-Dung, V., and Thierry, F. (2011). Fusion Between Laser and Stereo Vision Data For Moving Objects Tracking In Intersection Like Scenario. In *IV'2011 - IEEE Intelligent Vehicles Symposium*, pages 362–367, Baden-Baden, Germany.

Fremont, V., Florez, S. A. R., and Bonnifait, P. (2012). Circular targets for 3d alignment of video and lidar sensors. *Advanced Robotics*, 26:2087–2113.

Guo, C. X. and Roumeliotis, S. I. (2013). An analytical least-squares solution to the line scan lidar-camera extrinsic calibration problem. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2943–2948. IEEE.

John, V., Long, Q., Liu, Z., and Mita, S. (2015). Automatic calibration and registration of lidar and stereo camera without calibration objects. In *2015 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pages 231–237.

Li, Y., Ruichek, Y., and Cappelle, C. (2013). Optimal extrinsic calibration between a stereoscopic system and a lidar. *IEEE Transactions on Instrumentation and Measurement*, 62(8):2258–2269.

Lipu, Z. and Zhidong, D. (2014). A new algorithm for the establishing data association between a camera and a 2-d lidar. *Tsinghua Science and Technology*, 19(3):314–322.

Nez, P., Drews, P., Rocha, J. R. P., and Dias, J. (2009). Data fusion calibration for a 3d laser range finder and a camera using inertial data. In *ECMR*, pages 31–36.

Park, Y., Yun, S., Won, C. S., Cho, K., Um, K., and Sim, S. (2014). Calibration between color camera and 3d lidar instruments with a polygonal planar board. *Sensors*, 14(3):5333–5353.

Trawny, N. and Roumeliotis, S. I. (2005). Indirect kalman filter for 3d attitude estimation. *University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep*, 2:2005.

Wittmann, D., F., C., and M., L. (2014). Improving lidar data evaluation for object detection and tracking using a priori knowledge and sensorfusion. In *Informatics in Control, Automation and Robotics (ICINCO), 2014 11th International Conference on*, volume 1, pages 794–801. IEEE.

APPENDIX

- The quaternion is generally defined as

$$\bar{q} = q_4 + q_1 i + q_2 j + q_3 k \tag{14}$$

where i, j, and k are hyper-imaginary numbers and the quantity q_4 is the real or scalar part of the quaternion.
- To convert a 3D vector $\overrightarrow{p}$ to quaternion form we use

$$\bar{p} = [\overrightarrow{p} \quad 0]^T \tag{15}$$

-The product $\overrightarrow{p}_2 = R\overrightarrow{p}_1$ where $\overrightarrow{p}_1$, $\overrightarrow{p}_2$ are vectors, R is the rotation matrix and q its quaternion equivalent, can be written as follow

$$\bar{p}_2 = q \otimes \bar{p}_1 \otimes q^{-1} \tag{16}$$

where $\otimes$ presents quaternion multiplication, q^{-1} is the quaternion inverse defined as $q^{-1} = [-q_1 \ -q_2 \ -q_3 \ q_4]^T$.
-For any quaternions q_a and q_b, the product, $q_a \otimes q_b$ is defined as

$$q_a \otimes q_b \triangleq \mathcal{L}(q_a) q_b = \mathcal{R}(q_b) q_a \tag{17}$$

Where:

$$\mathcal{L}(q) = \begin{bmatrix} q_4 & -q_3 & q_2 & q_1 \\ q_3 & q_4 & -q_1 & q_2 \\ -q_2 & q_1 & q_4 & q_3 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix} \quad \mathcal{R}(q) = \begin{bmatrix} q_4 & q_3 & -q_2 & q_1 \\ -q_3 & q_4 & q_1 & q_2 \\ q_2 & -q_1 & q_4 & q_3 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix}$$

Also we have following properties

$$\mathcal{L}(q_a)\mathcal{R}(q_b) = \mathcal{R}(q_b)\mathcal{L}(q_a)$$
$$q_b^T \mathcal{L}(q_a)^T = q_b^T \mathcal{R}(q_a)^T$$
$$\mathcal{L}(q^{-1}) = \mathcal{L}(q)^T$$
$$\mathcal{R}(q^{-1}) = \mathcal{R}(q)^T$$

For more details, the interested reader is referred to (Trawny and Roumeliotis, 2005).