TARGET-FREE EXTRINSIC CALIBRATION OF A MOBILE MULTI-BEAM LIDAR SYSTEM

H Nouira, Jean-Emmanuel Deschaud, F Goulette

TARGET-FREE EXTRINSIC CALIBRATION OF A MOBILE MULTI-BEAM LIDAR SYSTEM

H. Nouira[a], J. E. Deschaud[a], F. Goulette[a]

[a] MINES ParisTech, PSL - Research University, CAOR - Center for Robotics, 60 Bd St-Michel 75006 Paris, France - (houssem.nouira, jean-emmanuel.deschaud, françois.goulette)@mines-paristech.fr

Commission III, WG III/2

KEY WORDS: LIDAR, Calibration, Mobile Mapping, 3D, Automatic, Computer Processing, Velodyne

ABSTRACT:

LIDAR sensors are widely used in mobile mapping systems. With the recent developments, the sensors provide high amounts of data, which are necessary for some applications that require a high level of detail. Multi-beam LIDAR sensors can provide this level of detail, but need a specific calibration routine to provide the best precision possible. Because they have many beams, the calibration of such sensors is difficult and is not well represented in the litterature.
We present an automatic method for the optimization of the calibration parameters of a multi-beam LIDAR sensor: the proposed approach does not need any calibration target, and only uses information from the acquired point clouds, which makes it simple to use. For our optimization method, we define an energy function which penalizes points far from local planar surfaces. At the end of the automatic process, we are able to give the precision of the calibration parameters found.

1. INTRODUCTION

Light Detection and Ranging (LIDAR) sensors are useful for many tasks: mapping (Nuchter et al., 2004), localization (Narayana K. S et al., 2009) and autonomous driving (Grand Darpa Challenge, 2007) are some of the applications in which such sensors are used. Recently, multi-beam LIDAR sensors have appeared: they are more precise and give point clouds with high densities of points. In order to give correctly geo-referenced data with sensors mounted on a mobile platform, additional information from exteroceptive and proprioceptive sensors are needed. Also, the calibration of the whole system is required; however, it is mostly done with calibration targets, and needs human intervention, as for example with the system presented in (Huang and Barth, 2009). The calibration can take some time, and it is difficult to evaluate the precision of the method. The whole process of acquisition and data geo-referencement is illustrated with figure 1, with the different representations of each acquired point.
In this article, we call calibration of the multi-beam LIDAR sensor its extrinsic calibration: it consists in finding the rigid transformation between the LIDAR sensor and the IMU, so that the data acquired by the sensor are correctly projected in the navigation reference frame.
The solution we propose is, after the acquisition, to estimate the parameters of the calibration that give the "best" - depending on some criteria - point cloud. We present an unsupervised extrinsic calibration method for multi-beam LIDAR sensors, which does not need any calibration target. We start with an initial calibration, which does not need to be close to the true one. With an iterative process, we look for "better" calibration parameters, which improve the quality of the point clouds and minimize an energy function we will define in section 3..This method does apply for any multi-beam sensor mounted on a mobile platform (a vehicle or a robot), and retrieves the six extrinsic parameters - three of translation and three of rotation - of the transformation between the sensor and the IMU.
This paper is organized as follow: in section 2., we present the state of the art concerning the algorithms for the calibration of multi-beam LIDAR sensors. Section 3. presents our calibration method. Section 4. shows some results obtained with our algorithm, and section 5. gives a conclusion to this paper.

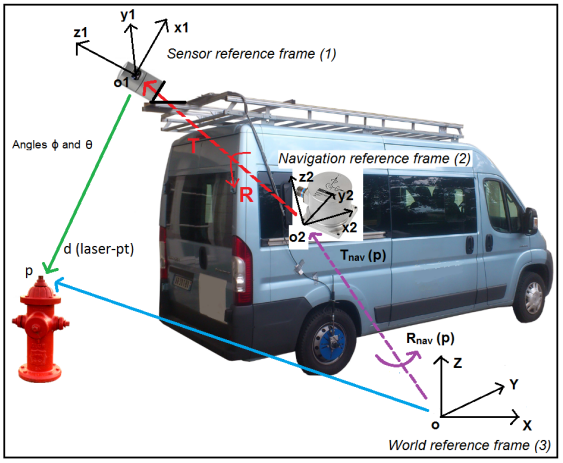

Figure 1: Geo-referencement of the data

2. RELATED WORK

Figure 1 illustrates the transformations between the different representations of an acquired point: this is our mobile mapping system, with the Velodyne 32-beam LIDAR sensor. The different steps for the data geo-referencement are the following:

- Raw data are acquired by the sensor mounted on the Vehicle. For a multi-beam sensor, these data are the distance of the point acquired to the sensor and two angles
- The raw data can be expressed in the sensor reference frame: this is done using the intrinsic calibration parameters.
- The extrinsic calibration gives the geometric transformation between the sensor and the IMU, and is needed to have coordinates registered in the navigation reference frame. The extrinsic calibration of a LIDAR sensor consists in finding

the transformation between the location on the mobile platform of the entire unit and the inertial measurement unit, also mounted on the mobile platform. There are six parameters to retrieve, three rotations and three translations. This is the calibration we want to optimize, and in this article, we will call calibration this extrinsic calibration.

- The data are geo-referenced by applying the transformation between the navigation reference frame and the world reference frame to these data.

Multi-beam LIDAR sensors appeared recently, and calibration techniques for this kind of LIDAR sensors already exist. If we take a look at the Velodyne sensor, some intrinsic calibration methods exist, like (Glennie and Lichti, 2010) and (Muhammad and Lacroix, 2010), which propose an optimization of the intrinsic parameters of the 64-beam version, or (Chan and Lichti, 2013), which proposes an intrinsic calibration for the 32-beam model.

In this paper, we are only interested in the extrinsic calibration of multi-beam LIDAR sensors. In (Zhu and Liu, 2013), the authors propose a method to optimize the 3 parameters of rotation into two steps: first, the roll and pitch angles by estimating ground planes, and after, the yaw angle by matching pole-like obstacles. This method is unsupervised and does not need a calibration target, as it uses only information from the data gathered; but, the limitation is that it only estimates the rotation parameters, and does not take into account the translation ones. In (Huang et al., 2013), the authors propose a full extrinsic calibration of a multi-beam LIDAR sensor, but they use calibration targets and infrared images to do this task. Also, in (Elseberg et al., 2013), the authors want to optimize the calibration of the whole system they use, which is composed of several LIDAR sensors. The energy function is a sum of Point Density Functions, which measures the compactness of their point clouds. They use an unsupervised and target-free method in post-processing, where an energy function they defined is minimized. The energy function was constructed in order to measure the compactness of the point clouds acquired. Finally, some approaches optimize both the intrinsic and the extrinsic calibration of a LIDAR sensor at the same time. This is the case in (Levinson and Thrun, 2010), where the authors present an intrinsic calibration and an extrinsic calibration method which uses the same energy function to minimize. For both calibrations, the authors chose an energy function which penalizes points that are far away from planar surfaces extracted from the acquired data. As in (Elseberg et al., 2013), this is a post-processing optimization, which is unsupervised and target-free. In (Levinson and Thrun, 2010), the authors start from an initial extrinsic calibration estimate, and iteratively compute values of their energy function by modifying the six extrinsic parameters in the neighborhood of the initialization. They use a grid search to optimize the parameters - for the minimization process, they alternatively test the translation parameters and the rotation parameters because of the difference of dimensionality -, and reduce the size of the neighborhood at each iteration. The main problem is that the minimization can be long if a high precision is required. Also, because the neighborhood is a discrete space, we possibly do not reach the optimal solution.

To optimize the calibration parameters of the multi-beam sensor, we use an energy function only using information extracted from the acquired point clouds.

No calibration target is used, and the process is unsupervised. The defined energy function is also minimized iteratively, as it will be explained in section 3.2.2. However, the differences with respect to existing methods are manyfold:

- First, the energy is defined as the sum of the squared distance of each points to the closest plane it should belong to, and its expected optimal (minimum) value is related to the global covariance of the point cloud noise.
- We also introduce in the energy weights which exploit the local planarity of data.
- Our method leads to a more accurate calibration for the point cloud.
- The numerical resolution is faster than existing methods, and is done in acceptable times.
- Finally, we also give an analysis of the precision obtained for the calibration parameters with the resolution. It will be explained with more details in section 3.3.

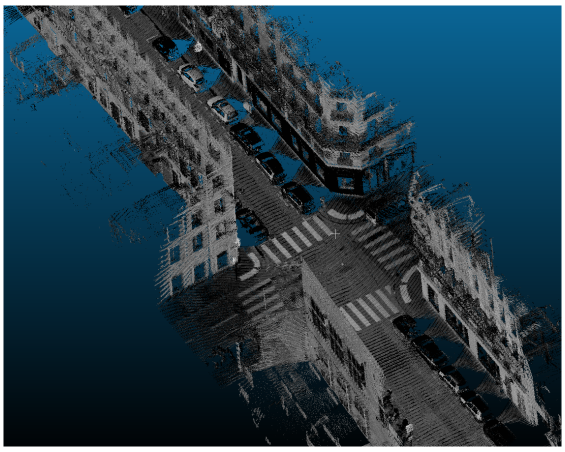

Figure 2: Example of a point cloud

3. PROPOSED OPTIMIZATION METHOD

We use a mobile mapping system to do our acquisitions, as presented in Figure 1. Many sensors are embedded on the vehicle, such as a BEI DHO5S odometer and an iXBlue LANDINS IMU to follow precisely the movement of the vehicle. We also have a Novatel FlexPak 6 GPS to retrieve the global position of the vehicle when possible, and finally a multi-beam LIDAR sensor, the 32-beam Velodyne, which is mounted on top of the vehicle, as shown on figure 1. The Velodyne sensor provides up to 700000 points/s, and covers a vertical field of view of 40° - from -8° to 32° - and an horizontal field of view of 360°. Also, we know the vehicle global pose at each control point, which allows us to project the acquired points in the global coordinate system. For our need, we only use the Multi-beam sensor to acquire data, and for geo-referencing the data, we use information given by the proprioceptive sensors and the GPS: the global position of the vehicle is retrieved by fusioning data from the IMU, the GPS and the odometer, which are measured during the mobile mapping. By taking a closer look at the point clouds, we can see that the acquired points tend to lie on surfaces, and most of these surfaces are locally planar: the lasers of the sensor reflect on many surfaces, like it is shown in figure 2. Indeed, during the motion of the vehicle, adjacent beams on the sensor will acquire points that belong to the same surface, but only if the extrinsic calibration of the sensor is good. This is what is illustrated in Figure 3, which can be seen as a side view of a planar surface, like a facade: with a wrong calibration, points acquired by neighbor beams will not be co-linear, where with a good calibration, lines of points acquired by close beams will overlap.

3.1 Definition of the energy function

To optimize the extrinsic parameters, we consider points which belong to planar surfaces and we exploit the previous observation, which is that these surfaces are not exactly planar with a

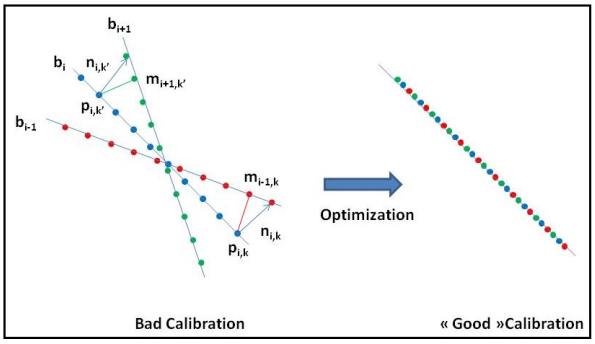

Figure 3: Side-view of a planar surface

wrong calibration. We start with an initial calibration, and only use information extracted from the point clouds. Eq. (1) gives the energy function we defined for the optimization of the extrinsic calibration parameters:

$$J(R,T) = K * \sum_{i=1}^{B} \sum_{j=i-N}^{i+N} \sum_{k} w_{i,j,k} * d^2_{i,j,k}(R,T) \quad (1)$$

where:

$$\begin{cases} K = \frac{1}{N_t - 6} \\ d_{i,j,k}(R,T) = n_{i,k} \cdot (p_{i,k}(R,T) - m_{j,k}(R,T)) \\ p_{i,k}(R,T) = R_{nav}(p'_{i,k}) * (R * p'_{i,k} + T) + T_{nav}(p'_{i,k}) \\ m_{j,k}(R,T) = R_{nav}(m'_{j,k}) * (R * m'_{j,k} + T) + T_{nav}(m'_{j,k}) \end{cases}$$

In equation 1, the other terms are:

- $K = \frac{1}{N_t - 6}$ is a parameter of normalisation of our energy. N_t is the number of paired points $p_{i,k}$. The energy we defined has a relation to physics: indeed, the energy unit is the cm^2 because it is a sum of the square of the distance between two points. Also, the distance measured with the quantity $(p_{i,k} - m_{j,k})$ represents the noise between the two points, which is equal to 0 in the ideal case: figure 3 illustrates this problem, where with a good calibration, there should be no noise and the energy should be close to 0. We suppose that this noise is independant for each point taken into account in the calculation of the energy J, centered, reduced and normal: with these hypothesis, the energy J follows a chi-squared distribution with K constraints. Energy J gives an estimate of the variance σ^2 of the point cloud noise when N_t is big enough, which is the case with our point clouds.
- B is a sample of the Velodyne sensor beams, with $B \subset [\![0; 31]\!]$
- N is half the number of neighbor beams to beam i taken into account
- k iterates on the considered points of beam i
- $w_{i,j,k}$ is a weight, which value is between 0 and 1 depending on the local planarity.
- $n_{i,k}$ is the normal to the tangent plane to point $p_{i,k}$. The normal is computed considering 2D information.
- $p_{i,k}$ and $m_{j,k}$ are respectively the k^{th} point of beam i, projected in the global reference frame and its nearest neighbor on beam j, also projected in the same reference frame.
- $p'_{i,k}$ and $m'_{j,k}$ are respectively the k^{th} point of beam i, projected in the sensor coordinate frame and its nearest neighbor on beam j, also projected in the same coordinate system.
- R_{nav} and T_{nav} are respectively the rotation matrix and translation vector from the navigation reference frame to the global reference frame. These matrix and vector depend on the time of the acquisition, thus they change from a point to another.
- Finally, R and T are respectively the rotation matrix and the translation vector from the sensor coordinate system to the navigation reference frame. R is represented with the three Euler angles, which are the calibration parameters of rotation roll, pitch and yaw, id est α, β and γ. The translation T, t_x, t_y and t_z represent the three translation parameters of the calibration we want to optimize. Thus, we have R(α, β, γ) and T(t_x, t_y, t_z).

3.2 Minimisation of the energy

3.2.1 Linear approximation

The energy function we use is penalizing points that are far from the local planar surfaces defined by points from the point cloud. If we have the optimum calibration parameters, the energy should be at a global minimum: this is not the case, because we start with an initial calibration different from the optimal one. We do not know how to solve energy J from equation (1), because the energy is not linear due to the calibration parameters of rotation. We change our problem by looking for little variations of the calibration parameters which reduce the energy J: starting from known parameters $(t_x, t_y, t_z, \alpha, \beta, \gamma)$, we look for small variations of the parameters $(\delta t_x, \delta t_y, \delta t_z, \delta\alpha, \delta\beta, \delta\gamma)$, which give the following result:
J(R($\alpha + \delta\alpha, \beta + \delta\beta, \gamma + \delta\gamma$), T($t_x + \delta t_x, t_y + \delta t_y, t_z + \delta t_z$)) < J(R($\alpha, \beta, \gamma$), T($t_x, t_y, t_z$))
We replace the matrix R($\alpha + \delta\alpha, \beta + \delta\beta, \gamma + \delta\gamma$) by an approximation R($\alpha, \beta, \gamma$) + $R_\alpha * \delta\alpha + R_\beta * \delta\beta + R_\gamma * \delta\gamma$, by using the fact that the variations we are looking for are small.

With this approximation in eq. (1), we get a linear least squares problem, with the following objective function to minimize:

$$S(\delta X) = \sum_{i=1}^{B} \sum_{j=i-N}^{i+N} \sum_{k} w_{i,j,k} * (D_{i,j,k} + C_{i,j,k} \cdot \delta X)^2 \quad (2)$$

where:

$$\begin{cases} \delta X = (\delta t_x \quad \delta t_y \quad \delta t_z \quad \delta\alpha \quad \delta\beta \quad \delta\gamma)^T \\ D_{i,j,k} = n_{i,k} \cdot \begin{pmatrix} T_{nav}(p'_{i,k}) - T_{nav}(m'_{j,k}) \\ + \left[R_{nav}(p'_{i,k}) * (R(\alpha,\beta,\gamma) * p'_{i,k} + T(t_x,t_y,t_z))\right] \\ - \left[R_{nav}(m'_{j,k}) * (R(\alpha,\beta,\gamma) * m'_{j,k} + T(t_x,t_y,t_z))\right] \end{pmatrix} \\ C_{i,j,k} = n_{i,k} \cdot \begin{pmatrix} \left[R_{nav}(p'_{i,k}) - R_{nav}(m'_{j,k})\right] * [1 \quad 0 \quad 0]^T \\ \left[R_{nav}(p'_{i,k}) - R_{nav}(m'_{j,k})\right] * [0 \quad 1 \quad 0]^T \\ \left[R_{nav}(p'_{i,k}) - R_{nav}(m'_{j,k})\right] * [0 \quad 0 \quad 1]^T \\ R_{nav}(p'_{k,i}) * R_\alpha * p'_{i,k} - R_{nav}(m'_{j,k}) * R_\alpha * m'_{j,k} \\ R_{nav}(p'_{i,k}) * R_\beta * p'_{i,k} - R_{nav}(m'_{j,k}) * R_\beta * m'_{j,k} \\ R_{nav}(p'_{i,k}) * R_\gamma * p'_{i,k} - R_{nav}(m'_{j,k}) * R_\gamma * m'_{j,k} \end{pmatrix} \end{cases}$$

The solution which minimizes the objective function (2) is the solution of the following linear system:

$$C \times \delta X = -V \quad (3)$$

with:

$$\begin{cases} C = \sum_{i=1}^{B} \sum_{j=i-N}^{i+N} \sum_{k} w_{i,j,k} * C_{i,j,k} * C^T_{i,j,k} \\ V = \sum_{i=1}^{B} \sum_{j=i-N}^{i+N} \sum_{k} w_{i,j,k} * D_{i,j,k} * C_{i,j,k} \end{cases}$$

3.2.2 Optimization approach

To find the optimal calibration parameters, we compute δX iteratively, until the calibration parameters converge. At each new iteration n+1, the calibration parameters are defined as follow:

$$\begin{cases} t_{x_{n+1}} = t_{x_n} + \delta t_{x_n} \\ t_{y_{n+1}} = t_{y_n} + \delta t_{y_n} \\ t_{z_{n+1}} = t_{z_n} + \delta t_{z_n} \\ \alpha_{n+1} = \alpha_n + \delta\alpha_n \\ \beta_{n+1} = \beta_n + \delta\beta_n \\ \gamma_{n+1} = \gamma_n + \delta\gamma_n \end{cases} \tag{4}$$

As it is presented in the next section 3.3, C is an important matrix which helps us get an estimate of the precision of the calibration parameters retrieved through optimization.
We start the optimization process with initial calibration parameters $(t_{x_0}, t_{y_0}, t_{z_0}, \alpha_0, \beta_0, \gamma_0)$, and at each iteration, to find the new variations δX, we minimize the objective function (2). We will see in section 4. that the initial calibration can be far from the true one, and that we do not need a precise estimate to start with. The complete algorithm is presented in algorithm 1.
The stop criterion we chose concerns the variation of δX during the iterations: the algorithm stops when $\|\delta X\|_{max}$ is under a threshold δ, or after a certain number of iterations.

3.3 Precision of the calibration parameters

In the state of the art of the calibration of mobile mapping systems, there is no value given to measure the precision of the optimized parameters: the authors generally compare their result point clouds to a ground truth, which can be hard and time-consuming, because the ground truth cannot be done with an automatic process; human supervision is needed to validate the process. In our method, we give values to measure the precision of the parameters retrieved with our optimization process: we have an estimate of the precision for each parameter, given by the following terms:

$$\begin{cases} \text{For translations: } \sigma_x(m) = \sqrt{(C^{-1})_{1,1}} \\ \text{For rotations: } \sigma_\alpha(rad) = \sqrt{(C^{-1})_{4,4}} \end{cases} \tag{5}$$

The remaining precisions are defined the same way for the translations and rotations parameters. C is the matrix defined in eq. (3), and its inverse $C^{-1} = Cov(\delta X)$. With the square roots, we have the precisions of each parameter.
These precision depend on the structure of the point cloud and the trajectory of the vehicle. For example:

- a point cloud with some features, such as turns or a variation of altitude, will give a better precision for the calibration parameters.
- a trajectory which goes straight, without changes, will not give a good precision for the parameters.

3.4 Validity of the calibration

We defined in the previous sections an energy function that, with an optimization process, should gives better calibration parameters for our points clouds. We will discuss in this section about the condition that validate a calibration obtained with our optimization process. The value of the energy J should be small enough, under a threshold: as said in section 3.1, our energy follows a chi-squared distribution with $N_t - 6$ constraints. A validation threshold at 97% is $3\sigma^2$, with σ^2 the variance of the point cloud noise. For example, for real data, the noise comes from the acquisitions, but also the IMU and the GPS which give the trajectory of the vehicle; with our mobile mapping system, we have a good precision, with a standard deviation for the noise around 5cm. It gives us a threshold of around 75 cm^2 for the value of energy J.

Algorithm 1: Iterative linear optimization for the parameters of the extrinsic calibration

Data: A point cloud, with an initial calibration, and a known trajectory
Result: A point cloud, with optimized calibration parameters
Reading and sub-sampling of the point cloud;
Initial extrinsic calibration;
repeat
  Global referencing of the points acquired with the actual calibration $(t_{x_n}, t_{y_n}, t_{z_n}, \alpha_n, \beta_n, \gamma_n)$;
  Selection of the points $p_{i,k}$;
  Construction of the pair of points $p_{i,k}$ from beam i and $m_{j,k}$, their closest neighbor from beam j;
  Computation of the normals to each planes at points $p_{i,k}$;
  Construction of equation (3), and resolution which gives the variations $(\delta t_{x_n}, \delta t_{y_n}, \delta t_{z_n}, \delta\alpha_n, \delta\beta_n, \delta\gamma_n)$;
until $\|\delta X\|_{max} < \delta$, *or* $n_{iter} \geq n_{max}$;
Saving of the modified point cloud with the optimized calibration;

4. EXPERIMENTS

4.1 Datasets

In our experiments, we used two types of data: synthetic and real point clouds. The difference between them concerns the way the point cloud is "created": for synthetic data, we simulated an acquisition in a urban area, with vertical planes as walls and an horizontal plane as the ground. For real data, we used our mobile mapping platform to acquire point clouds in a city. But, other than that, we use the same informations for both data. We have raw informations from the sensor, which is composed of:

- the position of the vehicle at each acquisition instant. This is the position of the IMU in the world reference frame, fused with other informations from proprioceptives sensors, such as the GPS and the odometer.
- the coordinates of each acquired point in the spherical coordinate system of the sensor reference frame.
- the "beam" which acquired each point, since we work with a multi-beam sensor.

These informations give us the position of the vehicle and its trajectory with a high precision: indeed, we only work on the calibration of the LIDAR sensor, and to have a well reconstructed point cloud at the end of the optimization, the trajectory has to be known precisely.

4.2 Implementation and algorithm parameters

The algorithm we presented was implemented in C++. The EIGEN library (Eigen library, 2014) was used for all the operations on matrices or vectors, and the FLANN library (FLANN library, 2014) (Fast Library for Approximated Nearest Neighbor) was used for the nearest neighbors search. The algorithms runs on a computer with a Windows 7 - 64 bits os, 32 GB of RAM and an intel core-i7 processor, with a clock up to 3.40 GHz.
Our algorithm was tested with synthetic and real urban data: for the synthetic data, the parameters were known precisely. For the real data, we didn't know the true calibration: for both data, we

	t_x(cm)	t_y(cm)	t_z(cm)	$\alpha(^\circ)$	$\beta(^\circ)$	$\gamma(^\circ)$
Initial calibration	-200.00	240.00	-150.00	5.00	-37.00	-5.50
Difference between the ground truth and the state-of-the-art optimization	**0.00**	400.00	-50.00	-0.25	**0.00**	-0.25
Difference between the ground truth and our optimization	**0.00**	**0.02**	**0.02**	**0.00**	-0.01	**0.06**

Table 4: Optimization of the calibration of synthetic point cloud #1

	t_x(cm)	t_y(cm)	t_z(cm)	$\alpha(^\circ)$	$\beta(^\circ)$	$\gamma(^\circ)$
Initial calibration	-200.00	240.00	-150.00	5.00	-7.00	-5.50
Difference between the ground truth and the state-of-the-art optimization	240.23	640.23	**190.23**	0.01	0.01	6.09
Difference between the ground truth and our optimization	**0.01**	**-0.13**	-200.00	**-0.01**	**0.00**	**-0.00**

Table 5: Optimization of the calibration of synthetic point cloud #2

	σ_{t_x}(cm)	σ_{t_y}(cm)	σ_{t_z}(cm)	$\sigma_\alpha(^\circ)$	$\sigma_\beta(^\circ)$	$\sigma_\gamma(^\circ)$
Synthetic point cloud #1	1.49	3.43	5.58	0.97	0.05	0.10
Synthetic point cloud #2	0.78	3.51	$1.00 * 10^{20}$	0.04	0.05	0.05

Table 6: Precision of the parameters found for the simulated data

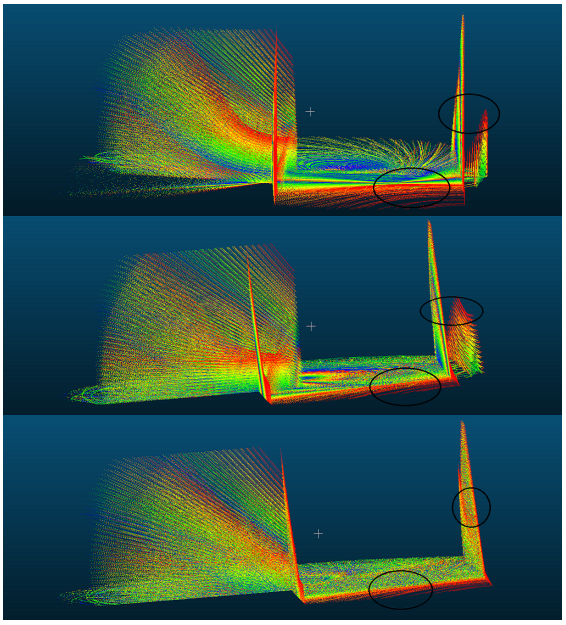

Figure 7: Synthetic point cloud #1: on top, with the initial calibration; middle, with a state of the art optimization; at the bottom, with our optimization. The three images have the same point of view.

started with an initial calibration arbitrarily chosen.
In our algorithm, we have some parameters to set. We start with sub-sampling the data about 1 point out of 2, because the point clouds have a high resolution and in order to reduce the computation times and the use of memory. The number of neighbor beams for a beam b_i was fixed to 4 (N=2). Concerning the weights $w_{i,j,k}$, a threshold of 20 cm was chosen for the maximal distance between a point $p_{i,k}$ and its nearest neighbor $m_{j,k}$ on the neighbor beam. We also considered that all the points belonged to planar surfaces, because the real point clouds presented in this section were acquired in a urban environment. Finally, the number of closest neighbors N_{cn} taken into account for the search of the neighbor $m_{j,k}$ of point $p_{i,k}$ was set to 100.
The parameters were fixed for all the tests which were done: different values were tested, but the ones presented give the best optimization results and computation times.

4.3 Comparison to the state of the art

We compared our approach and results with the optimization presented in (Levinson and Thrun, 2010), where the calibration of

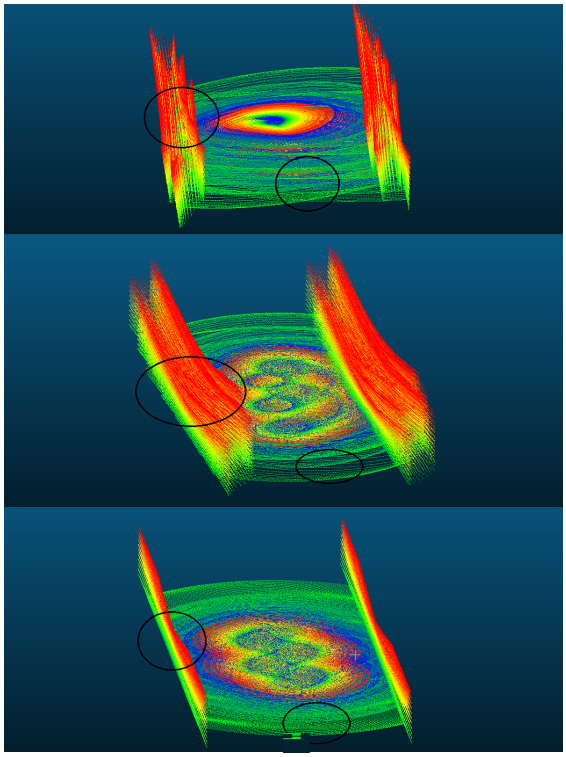

Figure 8: Synthetic point cloud #2: on top, before optimization of the calibration; middle, with a state of the art optimization; at the bottom, after our optimization. The three images have the same point of view.

a Velodyne sensor is also being optimized automatically. The energy function used and the optimization method are different: their method is a discrete search of the optimal parameters in the neighborhood of their starting calibration. They try several combination of parameters value to find the optimal calibration for the sensor, and repeat the process iteratively with reducing the space of search. They also separate the search for the translation parameters first, and then for the rotation parameters.
They set various parameters in their algorithm, which are:

- the size of the neighborhood for the optimization; it reduces with each iteration, until they have the needed precision for the parameters.
- a step of discretization - different or not from a parameter to an other -, which is the "number" of values tested in their neighborhood for each parameter.

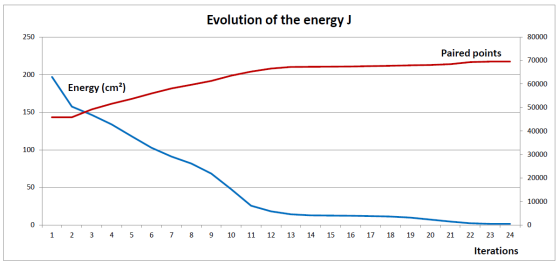

Figure 9: Evolution of the energy of synthetic point cloud #1

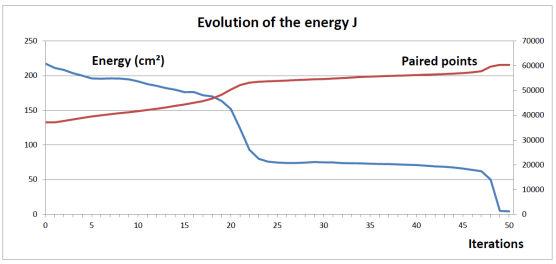

Figure 10: Evolution of the energy of synthetic point cloud #2

- the number of iterations, which determines the precision of the calibration parameters.

In section 4.4, we present a comparison between our algorithm and the one presented in (Levinson and Thrun, 2010). For these tests, we set some parameters for this optimization method: we used a step value of 5 for the grid search of each parameter, and our research was done between -3 and 3 meters around the starting values for the translation parameters, and between -7 and 7 around the starting values for the rotation parameters. We also did 10 iterations for the optimization: at each iteration, the neighborhood size was divided by two, and this number of iterations led us to a good precision for the parameters.

4.4 Results on synthetic data

The synthetic data are point clouds which simulate acquisitions in urban zones, created specifically for our tests: the scenes are composed of planes and the vehicle describes some trajectory.
In this section, we will present two synthetic point clouds, with different features. The point clouds were composed of around 4.5 Mpts. For these data, the true calibration is known, and is used to validate the results of our optimization. Several tests were performed on other data, giving similar results. The initial calibrations are set by adding a bias of some meters for the translation parameters and some degrees for the rotation parameters to the true calibration parameters. We will show two optimization results, both with a comparison between the results of our optimization and the one presented in (Levinson and Thrun, 2010).
The first tests we present are done on a point cloud composed of a ground and two orthogonal planes. There is also a variation of altitude, and the vehicle is making a turn. Figure 7 presents the same point cloud, with the same point of view, but with three different calibrations:

- Top, this is the point cloud with an initial calibration.
- Middle, the point cloud with the calibration results of state-of-the-art method.
- Bottom, the point cloud with the optimized calibration parameters given by our method.

With table 4, we have the initial calibration for the point cloud, and the difference after the optimization between the calibration parameters found with the two optimizations, and the true ones - which are known -. We can see that the results give the same conclusion as before: with the optimizations, we have parameters closer to the true ones, but we have globally the smallest difference between the parameters found with our optimization and the true parameters. The state-of-the-art optimization tests many values in a restricted neighborhood around the initial calibration. Since this is a discrete optimization, the best optimization is obtained when the initial calibration is close to the true one; but, for the tests we performed, the initial calibrations are far from the true ones, and with this method, the optimization can be stuck in a local minimum, which seems to be the case.
Figure 9 gives the evolution of our energy during the optimization process. The energy starts from a value of 197.04 cm^2, and reaches a value of 1.48 cm^2 with our optimization; at the opposite, the energy reaches a value of 7.87 cm^2 with the other method. For the optimization using Levinson method, we did 10 iterations for the optimization: after 10 iterations, our optimization gives an energy value of 5.05 cm^2, which is still smaller than with Levinson method. We have the same conclusion before, which is that our optimization gives better results than the state-of-the-art method. We also measure the precision of our calibration parameters retrieved with our optimization: they are given for the point cloud #1 in the first line of table 6. At the opposite, in the state of the art, no indicator of the precision of the results of an optimization is presented. As we could expect, we have a good precision for all the parameters, which goes along with the observations made with table 4, that the difference between the true calibration and the one retrieved with our optimization process is small. We can also notice that the parameters of rotation have a better precision than the ones of translations, which means that these parameters are correctly retrieved more easily.
We can also present the computation times of both optimizations method, for the total number of iterations: our optimization took about 1 minute and 15 seconds to give the result presented previously, where Levinson's optimization took nearly 36 minutes with the parameters chosen for the test.

We also present another test on a synthetic point cloud, with different features. Visually, on figure 8, we can see the improvements in the structure of the point cloud, which is the result of a closer calibration to the true one. The planes are correctly reconstructed in the end of the optimization.
Table 5 presents the initial calibration for the point cloud, and the difference after the optimization between the calibration parameters found with the optimization, and the true ones - which are known -. Like for the other point cloud, we can see that with our optimization, we have parameters closer to the true ones; we just have a difference for the translation parameter in z which is high. This problem comes from the structure of the point cloud and the trajectory of the vehicle: we have two parallel planes and the trajectory of the vehicle is an oscillation, which gives good precisions for nearly all the parameters. But, unlike the previous point cloud, there is no movement in z for the trajectory of the vehicle, which explains the problem of precision for the translation parameter in z.

Figure 10 gives the evolution of our energy during the optimization process. The energy starts from a value of 217.41 cm^2, and reaches a value of 4.18 cm^2; at the opposite, the energy reaches a value of 19.41 cm^2 with Levinson's method, which is also higher than with our method. After 10 iterations, our optimization gives an energy value of 13.85 cm^2, which is smaller than the energy with Levinson method. But, unlike the evolution of the energy of the previous point cloud, this time, the energy decreases by steps. It can be explained by the structure of our algorithm: as illustrated with 3, with our optimization process, we want to adjust

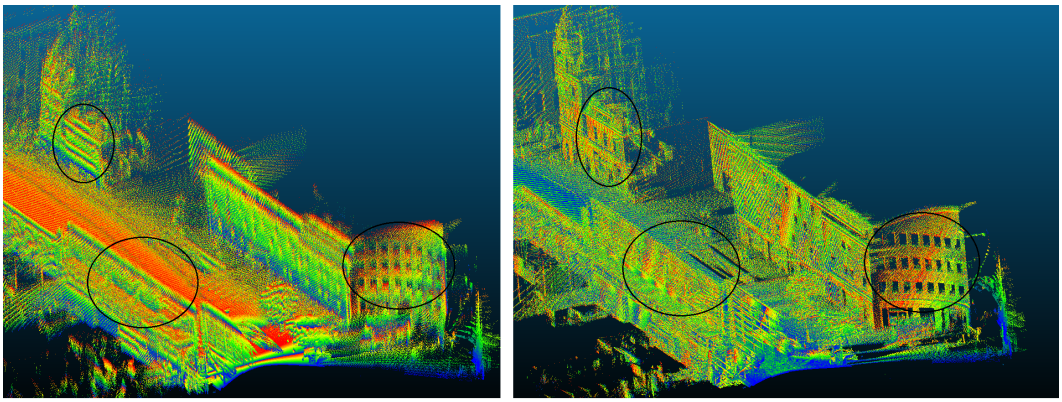

Figure 11: Real point cloud #3: left, before optimization of the parameters; right, after optimization. The two images have the same point of view.

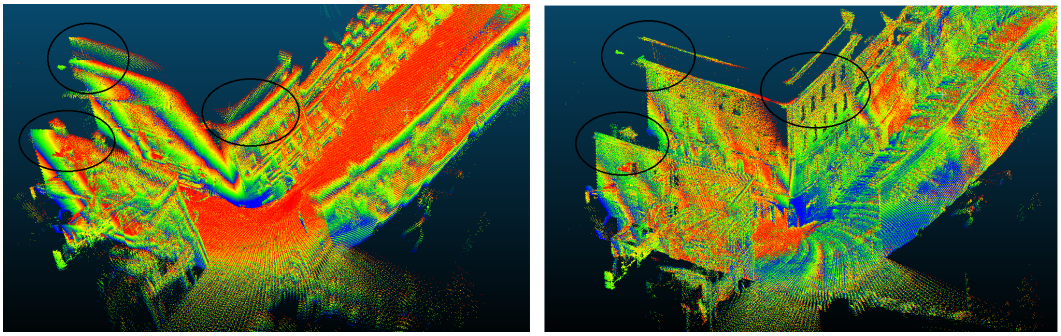

Figure 12: Real point cloud #4: left, before optimization of the parameters; right, after optimization. The two images have the same point of view.

the data acquired by neighbor beams of the sensor, which gives better calibration parameters. This problem is close to the adjustment of two datasets with an Iterative Closest Point algorithm: the energy J we constructed is also similar to the kind of energy minimized in an ICP approach. It has been proven in (Besl and McKay, 1992) that if for each iterations, the same number of pairs of points is taken into account for the calculation of the energy, the energy is always decreasing. But, in our case, the number of pairs of points taken into account increases with the iterations, as shown with the red curve, which explains the fact that our energy is not strictly decreasing: indeed, we take into account more pairs of point which are not relevant because the planes we want to adjust are too far from each other. When there is a peak in the number of pairs of points used, the planes are closer to each other, and the energy decreases. We also measure the precision of our calibration parameters retrieved with our optimization: they are given for the point cloud #2 in the second line of table 6. As we could expect, we have a good precision for all the parameters but the parameter of tanslation in z, which goes along with the observations made with table 5: the differences between the true parameters and the ones retrieved with our optimization process are small, except for the translation in z.
We also have a computation time for the total number of iterations of around 2 minutes for our optimization: it is longer than for the other synthetic point cloud presented, but still fast in comparison to the state-of-the-art optimization.

4.5 Results on real urban data

For the real point clouds, an acquisition was done in a city: because of the big size of the data, the whole acquisition was divided in several point clouds. Several tests were done, but the results are globally the same. We present two point clouds, which are made of 60 Mpts, and for which the true calibration is not known: to validate the results of our optimization, we use the conditions defined in section 3.4. The two point clouds represent different acquistions of the same city.
Figures 11 and 12 show the real point clouds: the circled areas give an idea of the improvements made with our optimization on the data. We can see that the façades are more planar after our optimization of the calibration parameters. Figures 13 and 14 give the evolution of the energy for each optimization, which are the blue curves: we can see that in the end of the optimizations, we have better values of the energies. But, the energies are not strictly decreasing; the red curves represent the evolution of the number of paired points taken into account in the calculation of energy Jat each iteration. The increase can also be explained the

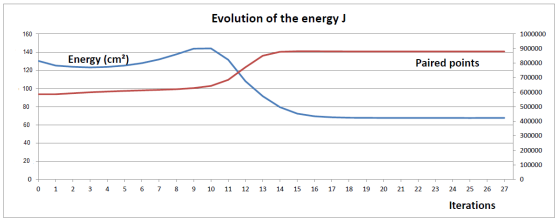

Figure 13: Evolution of the energy for real point cloud #3

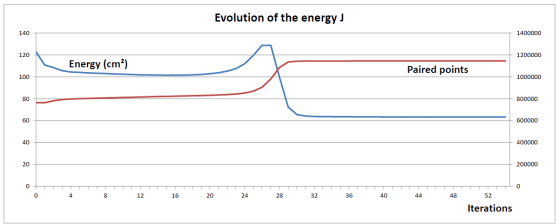

Figure 14: Evolution of the energy for real point cloud #4

	t_x(m)	t_y(m)	t_z(m)	$\alpha(^\circ)$	$\beta(^\circ)$	$\gamma(^\circ)$
Initial calibration	0.00	0.00	0.00	0.00	-45.00	90.00
Calibration after optimization	0.01	-1.37	-0.32	-8.11	-61.25	99.78

Table 15: Calibration of real point cloud #3

	t_x(m)	t_y(m)	t_z(m)	$\alpha(^\circ)$	$\beta(^\circ)$	$\gamma(^\circ)$
Initial calibration	0.00	0.00	0.00	0.00	-45.00	90.00
Calibration after optimization	-0.23	-1.39	-2.80	0.99	-62.03	91.79

Table 16: Calibration of real point cloud #4

	σ_{tx}(cm)	σ_{ty}(cm)	σ_{tz}(cm)	$\sigma_\alpha(^\circ)$	$\sigma_\beta(^\circ)$	$\sigma_\gamma(^\circ)$
Real point cloud #3	20.07	11.04	100.00	1.75	0.14	1.52
Real point cloud #4	2.41	1.81	47.51	0.34	0.05	0.29

Table 17: Precision of the parameters for the real point clouds

same way as in section 4.4.

For the point cloud #3, the energy starts at a value of 125.27 cm^2 and ends at a value of 67.67 cm^2. For the point cloud #4, the energy goes from 110.88 cm^2 to 63.37 cm^2. The values are acceptable, because we have noise from different sources, which increases the value of the energy: we are still able to improve the quality of the point clouds.

If we take a look at tables 15 and 16, we can see that the calibration parameters retrieved through optimization are different; but, as said before, the two point clouds are part of the same acquisition. The difference comes from the structure of the point clouds: indeed, point cloud #3 contains more elements which are not planar - like trees or poles - than point cloud #4. Thus, the optimization performs better on the second point cloud: table 17 gives the precisions of the parameters after optimization for both point clouds, which are different between the point clouds; but we see that the calibration parameters of point cloud #4 have in general a better precision than the ones of #3. Also, the energy of point cloud #4 is smaller than the one of #3: we can conclude that the calibration parameters of point cloud #4 are closer to the true calibration than the ones of #3.

The precisions are better for the rotation parameters than for the translation ones, which is the same observation than for the synthetic data: in general, the rotations are more easily correctly retrieved than the translations. We can also see that the precisions of the parameters of point cloud #3 are not really good: this is because the point cloud doesn't have many turns, and there are some non planar elements, such as trees, from which we take points into account for the optimization. This is the limit of our simplification hypothesis, which was that we considered all the points taken into account to belong to planar surfaces, such as stated in section 3.1.

We also have some computation times for the total number of iterations, which are for point cloud #3 and #4 respectively of 7min and 8min: we did not try to optimize these point clouds with state-of-the-art methods because, as shown in the previous section 4.4, the computation times are too high in comparison.

5. CONCLUSION

This paper presented a new and efficient optimization method for the calibration of point clouds. The optimization process is fully automatic, and can perform on small point clouds, or on point clouds with a large amount of points as well, in reasonable computation times. The results presented showed the efficiency and robustness of our algorithm, which only uses information from the point clouds.

REFERENCES

Besl, P. J. and McKay, N. D., 1992. A method for registration of 3d shapes. Transactions on Pattern Analysis and Machine Intelligence, IEEE, Vol. 14, pp. 239–256.

Chan, T. O. and Lichti, D. D., 2013. Feature-based self-calibration of velodyne hdl-32e lidar for terrestrial mobile mapping applications. The 8th International Symposium on Mobile Mapping Technology, Tainan, Taiwan.

Eigen library, 2014. `http://eigen.tuxfamily.org/index.php?title=Main_Page`.

Elseberg, J., Borrmann, D. and Nüchter, A., 2013. Algorithmic solutions for computing precise maximum likelihood 3d point clouds from mobile laser scanning platforms. Vol. 5(11), pp. 5871–5906.

FLANN library, 2014. `http://www.cs.ubc.ca/research/flann`.

Glennie, C. and Lichti, D. D., 2010. Static calibration and analysis of the velodyne hdl-64e s2 for high accuracy mobile scanning. Remote Sensing Vol. 2(6), pp. 1610–1624.

Grand Darpa Challenge, 2007. `http://en.wikipedia.org/wiki/DARPA_Grand_Challenge`.

Huang, L. and Barth, M., 2009. A novel multi-planar lidar and computer vision calibration procedure using 2d patterns for automated navigation. Intelligent Vehicles Symposium, Xi'an, China, IEEE.

Huang, P. S., Hong, W. B., Chien, H. J. and Chen, C. Y., 2013. Extrinsic calibration of a multi-beam lidar system with improved intrinsic laser parameters using v-shaped planes and infrared images. IVMSP Workshop, 2013 IEEE 11th, Seoul, South Korea.

Levinson, J. and Thrun, S., 2010. Unsupervised calibration for multi-beam lasers. International Symposium on Experimental Robotics.

Muhammad, N. and Lacroix, S., 2010. Calibration of a rotating multi-beam lidar. 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan.

Narayana K. S, K., Choi, S. and Goulette, F., 2009. Localization for mobile mapping systems, experimental results, analysis and post-processing improvements. 6th International Symposium on Mobile Mapping Technology, Sao Paulo, Brazil.

Nuchter, A., Surmann, H., Lingemann, K., Hertzberg, J. and Thrun, S., 2004. 6D SLAM with an Application in Autonomous Mine Mapping. Proceedings of the 2004 IEEE International Conference on Robotics and Automation, New Orleans, L.A. (April), pp. 1998–2003.

Zhu, Z. and Liu, J., 2013. Unsupervised extrinsic parameters calibration for multi-beam lidars. Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering, Paris, France pp. 1110–1113.