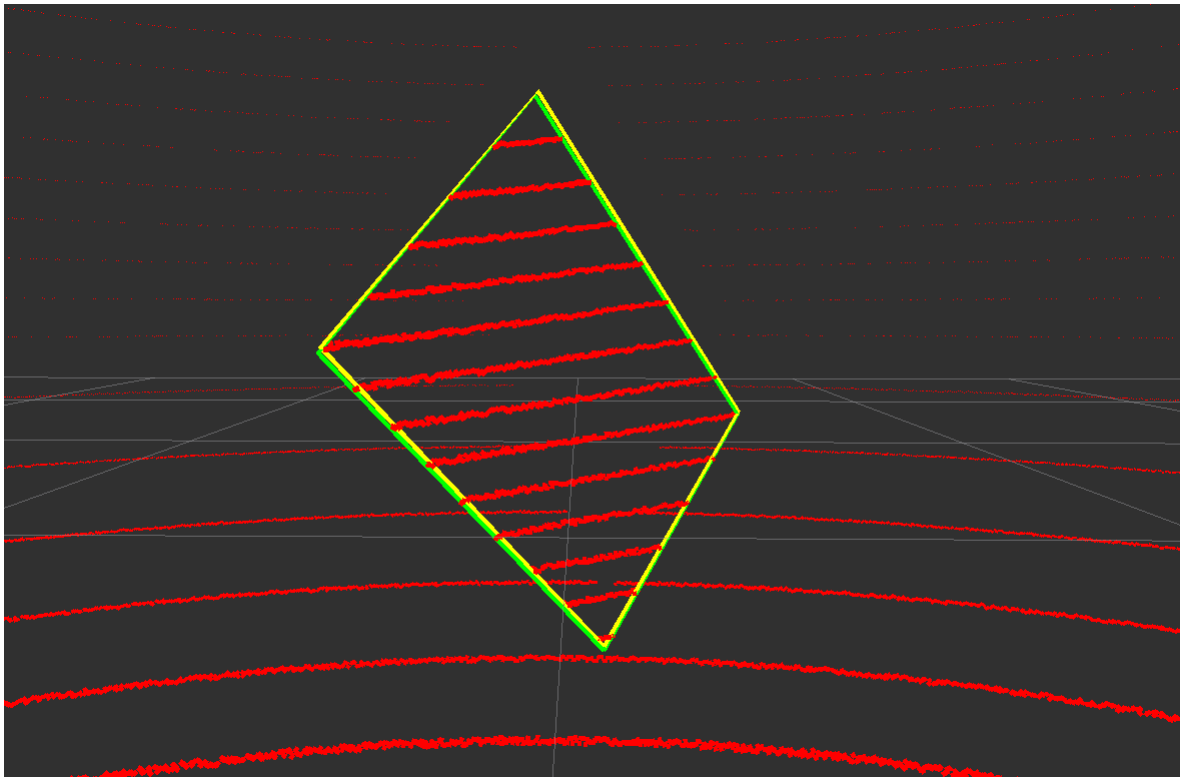
Internship in the Real Time Systems Group

Landais Erwann



Supervisor: Raphael Vosges
Tutor: Dr Luc JAULIN
Address: Applestrafe 9A, 30167 Hannover, Germany

September 30, 2019

Acknowledgement

I would like to thank Dr. Bernardo Wagner for giving me the opportunity to do an internship at the RTS Institute. I would also like to thank Professor Luc Jaulin for putting me in touch with the members of this university, thus making it possible to carry out this project; and Mr Raphael Voges, for his advice and the guidance he provided during my internship. Finally, I would like to thank all the staff of the RTS Institute, and especially Mrs Wilke and Mr Rauschenberg for their welcome, advice and help during my stay in Hanover.

Contents

List of Figures

Abstract

The purpose of this report is to describe the work done during my second year internship at Leibniz University in Hanover, in the Real-Time-Systems Institute. This three-month internship focused on the implementation of a procedure described by a scientific paper to merge data from one camera and several LIDAR sensors. The objective was to use the strengths of each of these sensors to obtain an accurate location of the environment of a device including these sensors. After recalling some notions on sensor fusion and describing the proposed procedure for performing this fusion, this report will outline the implementation chosen and the adjustments made in relation to the proposed procedure. Finally, this document will assess the accuracy of this fusion, using several data samples.

Résumé

Ce rapport a pour but de décrire le travail effectué durant mon stage de deuxième année d'école d'ingénieur au sein de l'université Leibniz d'Hanovre, dans l'institut Real-Time-Systems. Ce stage de trois mois s'est axé sur l'implémentation d'une procédure décrite par un article scientifique visant à fusionner les données d'une caméra et de plusieurs capteurs LIDAR. L'objectif était d'utiliser les points forts de chacun de ces capteurs pour obtenir un repérage précis de l'environnement d'un dispositif comprenant ces capteurs. Après avoir rappelé certaines notions sur la fusion de capteurs et décrit la procédure proposée pour effectuer cette fusion, ce rapport s'exposera l'implémentation choisie et les aménagements effectués par rapport à la procédure proposée. Enfin, ce document évaluera la précision de cette fusion, à l'aide de plusieurs échantillons de données.

Introduction

The Leibniz University Hannover is one of the leading universities in Germany. Composed of 9 faculties with a combined total of about 30,000 students, it also has 3,100 researchers in more than 180 institutes. These include the Real Time Systems Institute (or Instität für Systems Engineering / Fachgebiet Echtzeitsysteme). His field of activity is focused on mobile service robots and automation technologies. To this end, this institute mainly works on 3D perception, localization and path finding technologies for robots. He also works on discrete event modelling, real time systems programming of embedded control devices and network-based industrial automation systems. This institute is headed by Dr. Bernardo Wagner. The staff of this institute consists of about fifteen people working full-time. This staff consists of a secretary in charge of administrative tasks, a technician in charge of all the hardware tasks requested by the members of the institute (i.e. the production of 3D printing equipment or electronic components), and doctoral students. The institute also had about ten trainees during my internship, and took care of some projects with the students (for example on ordering a robot and following the line from it).

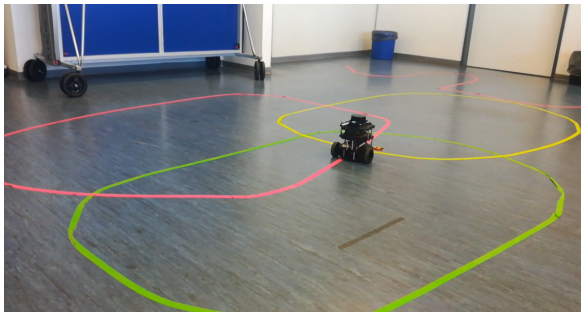


Figure 1: Example of student project

The projects on which the doctoral students of this institute work can be quite diverse. For example, one of the projects carried out was to create a tracking system for robots using sound using interval theory. Another was to merge the data between an infrared camera and a LIDAR, in order to accurately detect temperature variations in a metal plate. Finally, another project consisted in developing software to supervise the data of an autonomous car (%CPU, RAM used, WIFI signal...). Finally, Raphaël Voges, a doctoral student supervising my internship, was working on the fusion of sensors in order to improve the accuracy of the detection of autonomous systems. Indeed, one of the current challenges of robotics is to improve the ability

of autonomous systems to find their way through space, in order to increase the accuracy of their control and the reliability of these systems. This research topic, which has major issues concerning the autonomous car, involves improving sensor fusion systems. In the context of object recognition and spatial location, the fusion of LIDAR and camera is quite appreciated. As a matter of facts, a camera is a sensor whose price is quite low, but which is still able to easily differentiate objects in space. However, the disadvantage is that if the information provided by this sensor is reliable in the 2D plane of the image, this is not necessarily the case in the 3D plane of the camera. Therefore, there is always some uncertainty about the distance from the objects to the system, despite the calibrations. A LIDAR sensor, on the other hand, is a sensor that sends one or more laser beams in one direction. This sensor then receives their echoes, and by noting the time spent between transmission and reception of this laser can deduce the distance between the echo point and the sensor. The LIDAR then turns in another direction and restarts operation. A LIDAR performs these measurements at high speed, in order to obtain a 3D reconstruction of its environment quickly. This sensor has a high price, but can offer a high accuracy on the distance between the sensor and the detected objects. The disadvantage is that it is often very difficult to accurately identify the shape of objects, particularly because of the presence of noise. This complicates the differentiation of objects. Thus, these two sensors complement each other effectively, and are often used together to find their way through space. The objective is then to merge the data from them, to obtain precise information in real time about the location of the robot.

However, a number of conditions must be met before the data can be merged. First of all, the sensors must be calibrated, especially for the LIDAR, which is very sensitive to calibration errors. Then, it is necessary to be sure that the data are taken at the same time: no big delay or advance between two sensors, otherwise measurements will be absurd. Finally, it is necessary to know the homothety matrix (corresponding to a rotation and translation) between the sensors. This makes it possible, for example, to locate the points recorded by the LIDAR on the corresponding image, and thus to associate these points with the desired objects.

The objective of my internship was then to implement a method to perform this fusion: that is, to detect an object in a room and to know precisely its position. The final objective was to know as precisely as possible the position of the desired objects.

Chapter 1

State of the Art

There is different means to transfer the measurements from the sensors to the computer. One of those means is to use a software to transfer those data, called middleware. For example, ROS is a middleware which transfer data from the sensors to the computer in special buses, called topics. Those topics could be recorded into .bag files, in order to be exploited at any time and in order to facilitate their exploitation. Those data could also be recorded into other forms. Thus, the pictures taken by a camera could be recorded into png files. The measurements of a LIDAR could be recorded into a pcd file, which can express for each recorded point as much informations as required (location, altitude, laser used to record the point, ...).

1.1 Calibration of the sensors

Before trying to fuse the sensors, it is necessary to check if those sensors are well calibrated.

For a camera, calibrate means to obtain the relationship between the pixel of the picture and the location of the point in the real referential of the camera, expressed in meters. This relationship is based on the Pinhole Camera model. According to this model, the 3D points of the scene taken by the camera are projected into the plane of the picture. To do so, it is necessary to get intrinsic parameters of the camera : the focal lengths into the x and the y axis of the picture (fx and fy) and the location of the principal point of the camera (usually the image centre of the picture, expressed as (cx,cy)). In addition, lenses often have some distortion which can influence the accuracy of the position of the points on the pictures of the camera. This distortion can be evaluated thanks to several coefficients: 8 could be used, but generally 5 are enough to describe this distortion. Three of those coefficients (k1, k2, k3) are used to describe the radial distortion, whereas two of those coefficients (p1 and p2) are used to describe the tangential distortion. Some applications may also need to get the relationship between the referential of an object in a picture and the referential of the camera. This relationship corresponds by two transformations: a rotation and a translation. This is called homothety, and can be expressed by extrinsic coefficients, corresponding to the rotational and the translation matrix between those two referential (respectively R and t). Finally, the

general equations to go from the real referential of the camera to the plane referential of the picture could be expressed as presented by [1]:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = R * \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + t \tag{1.1}$$

$$x' = x/z$$

$$y' = y/z$$

$$r = (x')^2 + (y')^2$$

$$x'' = x' * (1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6) + 2p_1x'y' + p_2(r^2 + 2x'^2)$$

$$y'' = y' * (1 + k_1 * r^2 + k_2 * r^4 + k_3 * r^6) + p_1(r^2 + 2y'^2) + 2p_2x'y'$$

$$u = f_x * x'' + c_x$$

$$v = f_y * y'' + c_y$$

Where (X,Y,Z) are the coordinates of the point into the world referential system, and (u,v) are the coordinates of the point in pixels.

Respectively, the general equations to go from the plane referential of the picture to the real referential of the camera could be expressed as:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}^{-1} * \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \tag{1.2}$$

$$P_{img} = \begin{pmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$C_t = -\begin{pmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{pmatrix}^{-1} * \begin{pmatrix} p_{14} \\ p_{24} \\ p_{34} \end{pmatrix} = \begin{pmatrix} c_{11} \\ c_{12} \\ c_{13} \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = (-c_{13}/z') * \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} + C_t$$

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = R * \begin{pmatrix} x \\ y \\ z \end{pmatrix} + t$$

With (r11, ... r33) the components of R, (tx, ty, tz) the components of t, (X,Y,Z) the coordinates of the point into the world referential system, and (u,v) are the coordinates of the point in pixels.

For a LIDAR, it is necessary that each one of the laser beam used is calibrated, in order to be sure that all the objects detected by the LIDAR are accurately represented. To be sure that the distance recorded by each laser beam is accurate whatever the location of the LIDAR, five coefficients could be used. Two of those coefficients are used to correct the angle of the laser beam : the coefficient of vertical correction angle (phi) and the coefficient of rotational correction angle (tht). The three other coefficients are used to correct the distance between the sensor and the detected points: the distance correction offset, applied to the time of flight distance between the sensor and the detected point (dC); the horizontal offset coefficient, applied to the horizontal axis orthogonal to the laser beam (hOSC); and the vertical offset coefficient, applied on the vertical axis orthogonal to the laser beam (vOSC). Each set of those coefficients is different for each laser beam of the LIDAR. One of the techniques used to get those coefficients is presented by [2]. It consists to put the sensor into an empty room, to measure several distances between the sensor and the five walls of the room (the ground and the four walls of the room), and to calibrate one reference laser beam in order to match with the references taken. The other laser beams are then calibrated according to this reference laser beam, as all the other laser beams must be aligned with the reference laser on the different planes of the room.

1.2 Fuse of the sensors

Once the sensors have been calibrated, it is possible to focus on data synchronization. The first step is to make sure that this data is synchronized with time. To do so, it is necessary to have the time when the data was taken. There are several possibilities: either the sensor

automatically sends the time when the data was taken; there is then no work to do. Either the sensor does not send this information, and then we only have the time from which the information was received by the central unit. It is then necessary to use the documentation, the sensor characteristics, or even to create an additional system to obtain precisely when the data was taken. The sending of these times can be done via ROS topics. In this case, the time of receipt of the information by central unit and the time when the information was retrieved by sensor may be available in the same topic.

Once the data from the LIDAR and the camera are temporally synchronized, the next step is to find the relationship to move from the camera referential to the LIDAR referential, and vice versa. This change of reference frame is done via rotation and translation: finding the associated matrices therefore makes it possible to change the reference frame. Most of the time, the obtaining of these matrices between camera and LIDAR is based on the recognition of the characteristics of a painting with a chessboard on it: this type of painting is called checkerboard. It is necessary to know the number of squares, the size of the squares, sometimes even the size of the borders of this table. The objective is then to obtain the normal vector of the plane and some reference points of the checkerboard into the referential of the LIDAR and of the camera. At best, it could also be necessary to know the vectors representing the directions of the sides of the checkerboard, in the referential of the LIDAR and the camera.

Several methods exist to use the camera and LIDAR data to obtain these checkerboard characteristics. The one proposed by the article in [3] can be summarized as follows:

For images from the camera, the first step is to detect the checkboard on the image. Then, it is necessary to obtain the parameters of the characteristic equation of the plane, which can be expressed as $n * P + d = 0$. It is necessary to get the normal vector of the plane n and the constant d.

The next step is to remove the distortion on the picture and detect the image lines via LSD algorithm, which can detect all the lines into a picture. More details could be found in [4]. With this algorithm, it's then possible to detect the 4 lines surrounding the checkerboard, and to switch the line properties from the 2D referential of the image to the 3D referential of the camera, thanks to the equations of the "pinhole camera model". The objective is then to recover the plane parameters (normal vector, d) and the line parameters (the vector of the line, and a reference point on the line; to facilitate the identification of the points between the camera and the LIDAR, we will take the middle point of the lines here).

For LIDAR data, it is assumed that the area in which the plane is located is roughly known. The first step is then to reduce the whole detected scatter point to the area in which the plane is located. Then, it is necessary to detect the plane via the RANSAC algorithm. This algorithm is used to get parameter estimation of various mathematical objects (such as a plane or a line) from datas with a large proportion of outliers. More details could be found in [5]

The detected points should then be projected onto the plane. Afterwards, the scanned lines and their boundaries should be detected. The border points are then projected on the respective scanned lines. For the left and right borders, it is then necessary to detect the direction of each of the consecutive points on the border. These borders are subsequently divided at the level

of the most important change of direction. Next, the RANSAC algorithm is used to determine outliers on the border and to obtain 3D parameters of the lines constituting the border. The goal is to obtain the position of the points on the plane, the direction vectors and the normal vector, the mid-points on the plane and on the directions, as well as the points constituting the borders.

Finally, those data are used to get the rotational and the translation matrix between the two sensors thanks to a Levenberg Marquart algorithm. This algorithm consists in optimizing the solution of a system, by searching for variables that minimize the system equations. This requires starting from a satisfactory solution estimate, in order to allow the algorithm to converge to the right solution. More details could be found in [6]. For this first estimate of the solution, the rotation matrix is first determined from the normal vectors and boundary directions of the checkerboard, in each of the sensor referentials and for each of the poses. Then, the estimate of the translation matrix is obtained thanks to the equations of the plane and of the directions of each line into the referential of the camera, expressed for LIDAR points. This leads to a system which could be resolved thanks to a linear least-square algorithm. To avoid bias, the points used into those equations should be the middle points of the plane and of the boundary lines. Then, the equations used to estimate the translation matrix are used for the Levenberg-Marquart algorithm to refine the estimation of the rotational and of the translation matrix. All those equations are detailed in [3].

Chapter 2

Implementation of the method

2.1 The set of collected data

2.1.1 Simulated data

First of all, our algorithm is tested using simulated data to evaluate its robustness and accuracy. These simulated data were produced here via Gazebo. This software is able to simulate point clouds and photos with different checkerboard takes, and it is possible to add noise to test the robustness of the algorithms. Here, the simulated data is directly put into a.bag file. The topics used are general topics used for any type of sensor, allowing simple data manipulation. Thus, the topic used for the camera is of the sensor_msgs/Image type. For LIDAR, the topic used is sensor_msgs/PointCloud2. This topic allows to have access to the positions of the points but also indicates to which scanned line these points belong; it therefore allows to easily use the data contained in the.pcd files. For this experiment, 27 different poses were used.

2.1.2 Mesurements made during the internship

These tests were then completed through measurements made during the internship in a university room. Here, the topics from the sensors are into a raw format. Thus, for the camera: wfov_camera_msgs/WFOVImage is used. For LIDAR, velodyne_msgs/VelodyneScan is used. The sensors used were as follows:

*Camera: Pointgrey GS3-U3-23S6C-C. Some datasheet can be found in [7]

*LIDAR : Velodyne VLP-16 Puck : it is the same LIDAR than used in [3]. The 16 in its name indicates that this sensor has 16 rings. This means that for each measurement, 16 scanned dot rings are returned at different heights; each dot ring corresponds to a laser. The rings are then classified by increasing height.

Here, the poses are more difficult to perform, and not all orientations are possible. It is therefore not possible to limit oneself to simple diamond-shaped poses (which are the hardest to make): it is better to try to have more. We then had been able to get 34 poses during one

afternoon.

2.1.3 Data taken outdoors

Once these test phases were completed, the algorithm was used on a data set produced a few months before this internship, taken first in the room (in order to perform a calibration) before being released for circulation. The sensors used at the time were as follows:

*Camera: 2 Pointgrey GS3-U3-23S6C-C. Two are present on the assembly, in order to have two different angles of view. To obtain the transformation between camera and LIDAR, however, only one of the two will be used each time. Furthermore, it is associated with a device to measure the time when the camera is triggered by an electronic pulse.

*LIDAR: Velodyne HDL-64E S2 : Some datasheet could be found in [2]. This one has 64 rings.

2.2 Implementation of data fusion

The first step performed is the temporal synchronization of the data. It is performed for measurements directly from the sensors; i.e., as a topic wfov_camera_msgs/WFOVImage and as a topic velodyne_msgs/VelodyneScan. This merger was implemented by Mr. Vosges, who supervised this internship.

At first, some files and folders are created according to the requests of the user: .bag with new topics, and folders for.csv and .png files. Then, the data from the topics of the.bag are analyzed. These topics are of two types: topics containing data from the sensors (images, scatter points), and topics containing the times from which these data were recorded. Each data from the sensors is compared to the time topics; if there is consistency, the data is then saved in a.csv or.png file (depending on whether it is a scatter point or an image), or in a.bag file, depending on the user's request. In the latter case, the topics change: the LIDAR data are recorded in a topic called sensor_msgs/PointCloud2 for LIDARs, which allows to apply calibration coefficients to the measured points. For the camera, the topic becomes sensor_msgs/PointCloud2 for LIDARs, which allows easier image manipulation via OpenCV.

Since the .csv format is not the most suitable for LIDAR data manipulation, and in order to facilitate the manipulation of different poses while ensuring that they are the same for both sensors, I implemented a last step to this procedure which associates the .csv files with the .png files thanks to their associated time, and create .pcd files with the same name than the .png files. It is then possible to work on image and scatter points while being sure of the correspondence of the poses.

2.3 Exploit of the data

The choice of a convention is necessary to use this data. Indeed, all vectors (checkerboard normal vector, checkerboard boundary vectors) must be in the same direction with respect to the reference frame of the sensor studied, otherwise the measurements will not be usable. The convention adopted for this implementation is as follows:

*the normal vector is always directed as it is colinear to the axis from the sensor to the chessboard plane.

*the direction vectors rotate clockwise (anti-trigonometric)

*the lines are classified from bottom to top left, then from top to bottom right; 1st lines on the left, 2nd lines on the right.

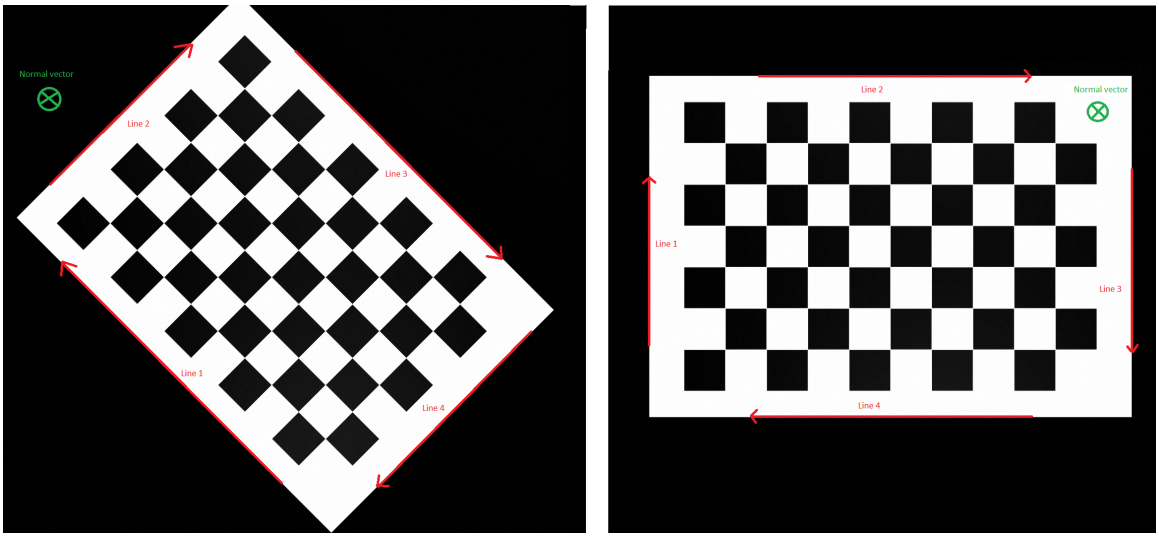


Figure 2.1: Convention adopted for parallel and non parallel lines

This convention is adopted to correspond to the checkerboard boundary recovery method proposed by the article for scatter points from LIDAR.

2.3.1 Analysis of the pictures

First of all, chessboard recognition is done using the already coded OpenCV methods; these methods are able to detect the corners of the squares present on the chessboard, and to note their position in pixels. The calibration matrix and distortion coefficients are also obtained with the calibrateMatrix function present in OpenCV. To do this, the corners of the chessboard are associated with coordinates associated with the reference frame of the checkerboard plan. For example, the top left corner may correspond to the point (0,0,0,0) ; etc...

With different poses, it is then possible to obtain an approximation of these different matrices. It may also be possible to obtain the rotation and translation matrix between the checkerboard marker and the camera marker for each of the poses treated. It should be noted that the corners of the checkerboard squares must remain reasonably distinguishable. This

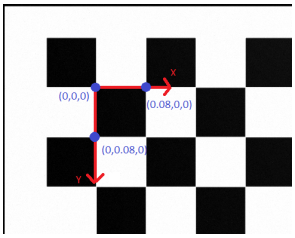


Figure 2.2: Coordinates in the reference frame of the checkerboard

implies that there is no reflection on the painting, and the table must be large enough on the image so that the corners of the squares can be distinguished.

It is then possible to proceed to the study of the poses that will be used to obtain the translation and rotation matrix between the camera and the LIDAR. The first step is to obtain the transformation matrix (rotation and translation) between the image reference frame and the camera reference frame for the currently processed pose. This is obtained using OpenCV's solvepnp method. The normalized normal checkerboard vector can be acquired once the rotation matrix is obtained, via the 3rd column of the rotation matrix. The next step is to recognize the boundaries of the table; more precisely, the directions of the checkerboard boundaries and its midpoints. To do this, it is first necessary to determine the corners of the board. The chess corners detected by the solvepnp method are used for this purpose. However, it is not sure that these corners are sorted so that they comply with the given convention. To solve this issue, the corners of the chessboard are associated with references corners on the picture, determined with the maximum and minimum on x and y of the corners of the chessboard and the angle between x-axis and the lower extreme corners of the chessboard is large enough. This association is done via distance measurement between these corners and the reference corners. The corners of the chessboard correctly classified are then used to obtain the mid-point of the checkerboard.

2.3.1.1 Acquisition of the boundaries of the checkerboard

There are then several methods to obtain the limits of the table.

The one proposed by the article studied is to use the LSD algorithm, which makes it possible to recognize the lines on the image to identify the limits of the table. The steps implemented to allow the use of this algorithm are then the following:

- detect all lines present in the processed image

- delete the lines detected on the chessboard. These can be obtained by knowing the position of the corners of the table

- complete the lines that should have been complete

- delete lines that are too small (which may correspond to poorly deleted lines on the chessboard)

- select the lines closest and most parallel to the chessboard boundary lines. These lines then correspond to the checkerboard limits.

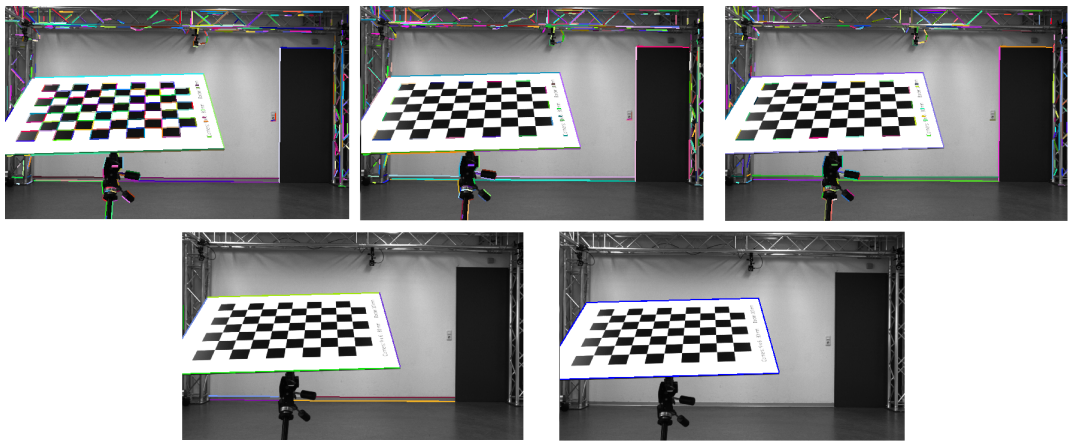


Figure 2.3: Successive steps in the use of the LSD algorithm

As these lines are very generally incomplete, it is then necessary to look for the point of intersection between the lines. These intersection points then correspond to the corners of the table, from which it is possible to obtain the midpoints of each line.

The advantage is that this method does not require information on the size of the checkerboard borders. The disadvantages are that many treatments are necessary for the checkerboard limits to finally be detected. In addition, this algorithm does not always work: some limits are sometimes not detected, or are incomplete. Finally, the interest of using such an algorithm may be questionable: since the size of the chessboard squares is necessary to obtain the rotation and translation matrix between image and camera repository, it may be easier to measure both the square size and the size of the checkerboard borders.

The other method to obtain the checkerboard limits is as follows: once the chessboard is obtained, by knowing the size of the table borders in the real world, it is possible to obtain the positions of the corresponding points at the corners of the checkerboard. Indeed, it is enough to give coordinates of the desired points in the checkerboard reference frame (in m), then to make transformation from table reference frame to frame of the image plane, expressed in pixel (thanks to pnp model equations).

The disadvantage of this method is that it requires precise knowledge of the size of the checkerboard borders. However, this one is very effective with good calibration, and is much simpler than the method using the LSD algorithm.

2.3.1.2 Acquisition of the directions of the boundaries

Once the sides of the table are obtained, it is possible to obtain the vectors of the table directions. To do this, we first check that the checkerboard corners obtained in the previous step are in the right direction; for this, we perform the transformation for each corner of the checkerboard from the reference frame of the image plane (in pixels) to the reference frame of the checkerboard (in m). The distance between these corners and the corners of the chessboard is then measured in the table's reference frame. Finally, each corner is associated with the

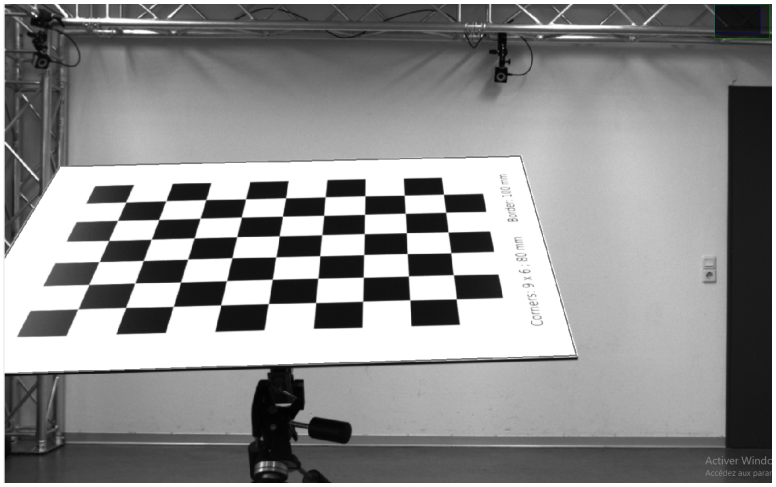


Figure 2.4: Detection of the corners of the checkerboard thanks to the pnp model equations

nearest extreme corner. From these corners set in the right direction, we then obtain the normalized direction and the mid-point of each line. Finally, we check that the vector of the table is in the right direction (positive z axis). It is then possible to send the normal vector and the mid-point of the checkerboard as well as the vectors and the mid-point of the checkerboard borders.

2.3.2 LIDAR data recovery

The manipulation of LIDAR data is based on the use of the ROS PCL library. Since this does not allow to associate to each point the laser that was at the origin of it, it is necessary to use another object to store this information during the manipulation of this data.

To do this, the first step is to determine all the scanned lines present on the studied pose, and to associate the detected points with the corresponding scanned line. The method used in this implementation is a list of lists, where each sub-list corresponds to a laser. The next step is to apply a filter corresponding to the area in which checkerboard is located, to eliminate unnecessary points. Depending on the detected area, it may be necessary to rotate the reference frame, to avoid a grimbal lock problem when detecting the rotation matrix between the two sensors studied. Once this area has been selected, it is then necessary to locate the plan of the checkerboard within it. This is done using the RANSAC algorithm, which is able to locate a plane in an area and give an approximation of the normal vector of that plane. The method performing this is contained in the ROS PCL library, and depends on an error coefficient (threshold) to detect which points present in the area can be considered as belonging to this approximation of the plane. This coefficient must be chosen carefully, to ensure that the plane and normal vector derived from the algorithm are consistent with reality.

Once this plan is detected, it may be possible to project the points on it as proposed by the article. However, this choice is not always interesting if the detected plane contains too many outliers; in this case, the normal vector may be distorted, and will then distort the real position

of the points. Once the plan is detected, it is possible to obtain the boundaries of the plan. The first step is to obtain the axis between the sensor (thus the origin of the point cloud) and the plane, as well as the axis perpendicular to the latter, with respect to the sensor reference frame. These two axes are obtained by considering that the area chosen for the detection of the plane is centered around it.

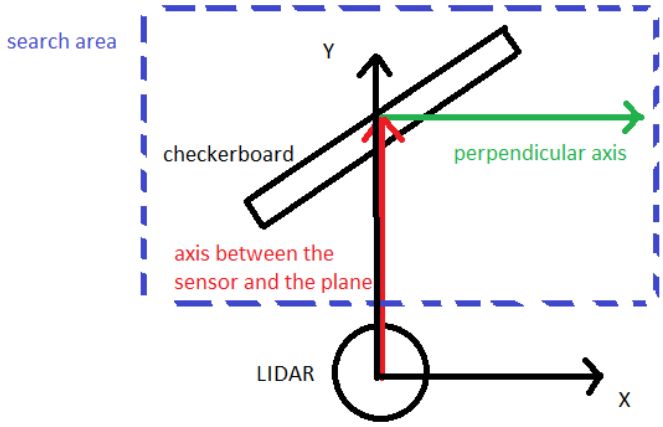


Figure 2.5: Drawing of the different axis used

The detection of these axes allows the points detected to be rearranged along the axis perpendicular to the axis between the sensor and the checkerboard. These points are rearranged so that the points considered to the left of the checkerboard correspond to low values of this perpendicular axis, while the points considered to the right of the checkerboard correspond to high values on this perpendicular axis. The lines selected in the previous step are then reclassified by increasing height.

Before determining the checkerboard's border directions, it is necessary to try to eliminate as much noise as possible from the detected lines. The first step for this is to eliminate the noise lines, corresponding to lines outside the checkerboard, that would have been selected when the checkerboard plane was detected; or lines being partly on the checkerboard, and partly on objects outside the checkerboard. These noise lines are identified by measuring the median difference between the distance from the boundaries of each consecutive line; therefore, from distance to left and distance to right, measured along an axis perpendicular to the checkerboard axis. If the distance difference is very superior to the median distance, an anomaly is detected. Several criteria are then used to determine which line should be deleted, depending on the height of the line from which the anomaly was detected, or the median distance. However, the number of lines deleted by this method could be quite huge and delete a lot of correct lines before finally deleting the incorrect line. Therefore, from a certain number of deleted lines it is estimated that it is no longer possible for there to be any noise lines left among the detected lines, and then we proceed to the next step of the algorithm.

To complete this suppression of noise that can interfere with the detection of checkerboard boundaries, we then try to remove lines whose boundaries would be inconsistent with the

rectangular shape of the checkerboard. Indeed, we know overall the shape that the checkerboard must have either a diamond or a square. We can therefore analyze each side of the checkerboard and delete the successive lines whose border points would not respect this shape. Finally, the lines are again classified by increasing height.

The last step is to separate the left and right checkerboard boundaries in two. The technique presented in the article is to take the biggest change of direction on the studied border, then separate it in two. The problem with this technique is that it assumes that the checkerboard poses are permanently in the form of "diamonds"; that is, that there are always at least 4 detectable directions. However, this limits the number of possible exposures, which can then be a problem in determining the precise transformation between the two sensors. However, taking into account all the biggest changes in direction at the borders, with a certain tolerance, lead to false detection of change of directions due to the noise remaining after filtering.

Another technique was then tried. This consists first of all in detecting the direction of a set of points; then, in obtaining the new direction after the addition of the next point. The difference between these two directions is then evaluated using the vector derived from the cross product between these two directions. If the angle and norm of this vector is low enough, then the point is considered to be in the continuity of this line; otherwise, a new line is created. This technique makes it possible to judge the relevance of a change of direction according to two criteria, which makes it possible to refine the detection of the change. In practice, this detection has proved to be quite random, making it difficult to determine coefficients from which a change in direction can be confirmed.

The technique finally adopted is as follows: each time a point is added, the equation of the line without this point is searched. The equation of the line with this point is then determined, and the difference for each point on the line between the point coordinate and the theoretical coordinate of the point is evaluated from the line equation. If the standard of these deviations is below a certain tolerance, then it is assessed whether the angle and standard between the old and new direction is below a certain tolerance. Here, the norm and the angle evaluated are inversely proportional: it is then possible to deduce a curve from it, making it possible to determine from which values it is possible to determine that a change of direction takes place.

Although more resistant to noise than the methods previously presented, this one sometimes contains some errors. A last filtering step is necessary, depending on the number of detected direction changes (corresponding to as many potential borders as have been detected). If more than 4 borders are identified, it means that one border has been detected too many to the left or right. The algorithm then identifies the border with the fewest points, and deletes all points and scanned lines associated with it, until it reaches a situation where two borders have been detected to the left and right of the checkerboard.

Once an adequate number of boundaries are detected, a RANSAC algorithm is applied to each set of points constituting these boundaries to infer the vector of each of these directions. If this algorithm detects that two consecutive lines are co-linear, it merges them together. It is then necessary to obtain the midpoints of the lines and the plane. If 4 borders are detected, the line directions are used to find the border crossing points: from these intersection points,

the midpoints of each direction are obtained. If only 2 borders are detected, this means that we are in the case where one of the checkerboard axes is parallel to the LIDAR. Then, the first and last scanned line are temporarily used to complete the borders, and to obtain an approximation of the intersection points and then an approximation of the mid-point of each line. Once all the midpoints have been obtained, we can finally deduce the mid-point of the plan.

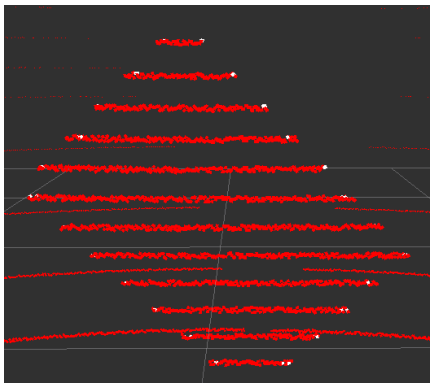


Figure 2.6: LIDAR scatterpoint with boundaries detected (in white)

It is then possible to send all the points present on the plan and the points present on the checkerboard borders. These are differentiated by an integer, indicating the line at which these points should be used. Then the vectors of each direction are sent with the mid-point of each border and the normal vector, with the mid-point of the plane. If a border has been approximated or has not been detected, an error point (corresponding to a point with -1 at all its coordinates) and an error vector are sent (in which all coordinates are set to 0 except the z-coordinate, whose value corresponds to the boundary that should have been detected). The number of LIDAR poses processed is also sent.

If the LIDAR data is very damaged, another mode can be started for the algorithm: it goes through all the steps until the noise lines are removed. Once these have been deleted, the points kept on the plane and the normal vector are sent directly. The mid-point of the plane is then estimated from the unnoised lines. This reduces the accuracy of the characteristics of the detected pose. However, this avoids having to detect the checkerboard boundaries, which are very difficult to obtain with precision for LIDAR.

2.3.3 Fusion and exploitation of sensor data

Once all the checkerboard data has been recovered from both the camera and LIDAR, it is necessary to collect it. We use ROS for this. Two nodes are created for this purpose:

- a node that handles the processing of images from the camera. The latter also retrieves the estimated homothety matrix and the data from the LIDAR, to project these points onto the camera poses and obtain visual confirmation of the quality of this estimate. It is called camera_detect.

- a node dealing with the processing of data from LIDAR. It also retrieves the data from the camera processing, to merge all this data and obtain an estimate of the homothety matrix between the two sensors. It's called cameraLIDARdir.

These nodes only recover data from.png and.pcd files in order to select the desired poses. The data is sent between each node via float64multiarray topics.

All data is recorded in an Listen object, which gathers all this data in two lists : a matrix list list list list for the LIDAR, called all_datas_LID; and a matrix list list for the camera, called all_datas_Cam. The steps for collecting the data are as follows :

The first is the recovery of the OpenCV package data. We recover as many as the poses sent have been indicated. It is assumed that all directions are detected for each pose (which is very likely, especially if the detection of directions takes place via the method using the checkerboard borders). All the information about these poses is then placed in the all_datas_Cam list.

The second step is the recovery of the data from the LIDAR. To do this, we first check that there is no direction vector or point corresponding to an error. If one of the detected borders corresponds to an error, it is indicated within the Listen object for the associated pose. All the information concerning these poses is then placed in the all_datas_LID list.

Once this data is collected, the next step is to combine the poses from the camera and from the LIDAR, to verify that all the necessary information (border vectors for example) has been retrieved by each sensor. If this is the case, the algorithm continues. If information is missing for a pose, the algorithm only retrieves the directions common to both sensors. The points corresponding to the directions not detected by LIDAR are then considered as points of the plane, and the vectors associated with the missing directions of the concerned poses are deleted for each sensor. Depending on the user's choice, it is also possible to recover only normal vectors as soon as a direction is missing for a pose. In this case, all the points detected by the LIDAR are considered as points of the plan, not associated with the checkerboard boundaries.

The next step is then to obtain a first approximation of the rotation and translation matrix, based on the equations presented in the article. This is done using the methods available in ROS libraries. From this approximation, we can determine if the measurements obtained are correct or not; indeed, the Levenberg-Marquart algorithm being based on the same equations, it leads roughly on the same solution. From these approximations, we can then create the variables necessary to use the Levenberg-Marquart algorithm. Finally, this algorithm is used to refine the approximation of the rotation and translation matrix. In order to limit the number of operations, the parameters optimized by this algorithm are the translation matrix coefficients, roll, pitch, and yaw. This allows only 6 parameters to be estimated (instead of 12 for each coefficient of the rotation and translation matrices).

Once the rotation and translation matrices have been refined, it is possible to send them to the camera node via a float64multiarray message. To do this, first the rotation and translation matrix is sent, then LIDAR points of each pose separated by a marker. All this information is retrieved in a lid2pic object. This displays the photos of all the poses with associated LIDAR points projected on them.

Chapter 3

Evaluation of the results

There are different methods to evaluate the accuracy of the rotation matrix and translation. This accuracy can be tested by projecting information from the LIDAR repository to the camera repository; but also by projecting information from the camera repository to the LIDAR repository.

The transformation from the LIDAR repository to the camera repository is done in the same way as expressed in the previous section. A color code is also associated with the points, in order to express their distance from the sensor: the closer the color is to blue, the further away the point is from the sensor.

The transformation from the camera reference frame to the LIDAR reference frame is done as follows: the normal vector and the points on the sides of the checkerboard are recovered on the studied pose. These are expressed in the LIDAR reference frame, linked together and then represented on Rviz at the same time as the point cloud studied.

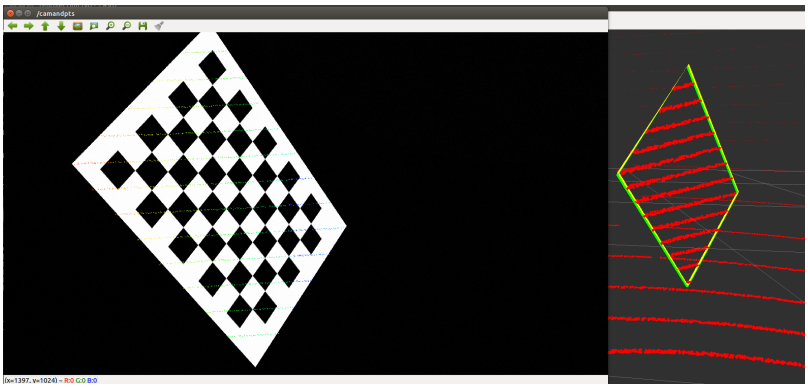


Figure 3.1: Transformation from LIDAR to camera (left), and from camera to LIDAR (right)

This verification is done using two ros nodes : one node to get the LIDAR datas and return the points to the area chosen by the user, called , and another to obtain the image taken by the camera and associated with these points, in order to draw these points on the image. If necessary, the latter node can also return the checkerboard plane to the LIDAR sensor

repository. This then allows the.bag files to be used to check the accuracy of the homothety matrix; therefore, all available data can be used.

3.1 Estimation of accuracy difference from simulation

3.1.1 With only one pose

According to the article, a single pose may be sufficient in theory to obtain a satisfactory estimate of the homothety matrix. To verify this statement, we take a pose in which the 4 borders of the checkerboard are clearly visible among the simulated data. The maximum difference between the theoretical coordinates and the coordinates discovered in practice is then checked. This deviation is indicated as a percentage deviation from the theoretical value for the rotation matrix; for the rotation matrix, this deviation is indicated as a degree deviation from the axis with the largest deviation. The homothety matrix was obtained in three different ways: one according to the article method, i.e. with normal vectors and vectors of the plane boundaries. Another was carried out only with the normal vectors of the plans, however following the steps of the article. Finally, the last one was performed with normal vectors and an approximation of the mid-point of the plane, obtained after the removal of all noise lines.

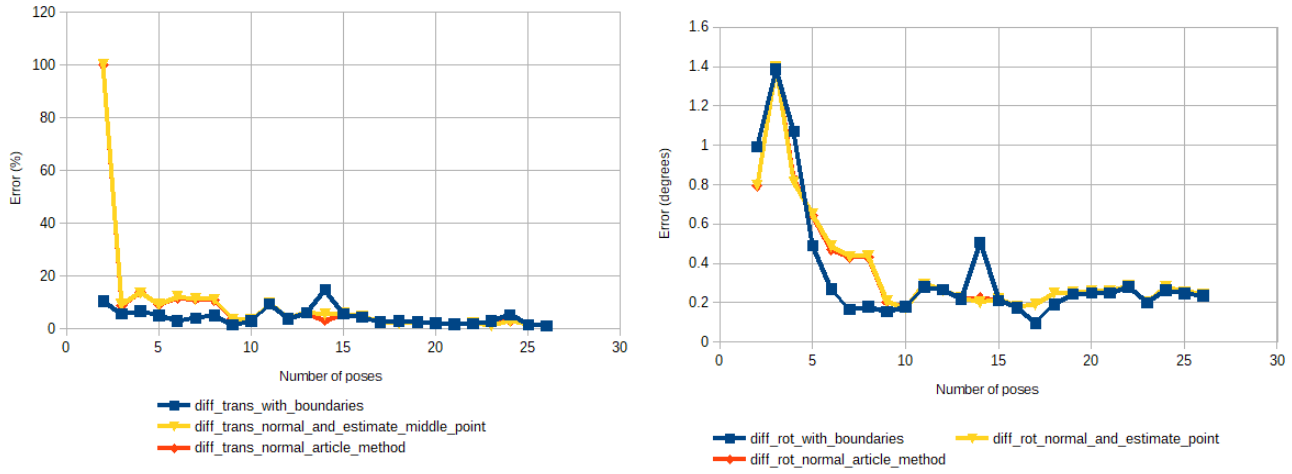


Figure 3.2: Maximal errors in function of the number of poses

It can therefore be seen that the first two first methods are not at all effective in obtaining the homothety matrix from a single pose. On the other hand, the method developed by the article allows to obtain directly a reasonable approximation of this matrix after a limited number of poses. However, this result is not yet satisfactory: thus, the maximum difference encountered for the material method remains significant (greater than 10% of the theoretical value).

3.1.2 With more poses

A single pose therefore does not seem sufficient to obtain a satisfactory estimate of the homothety matrix. By gradually increasing the number of poses, the following deviations from the theory are obtained:

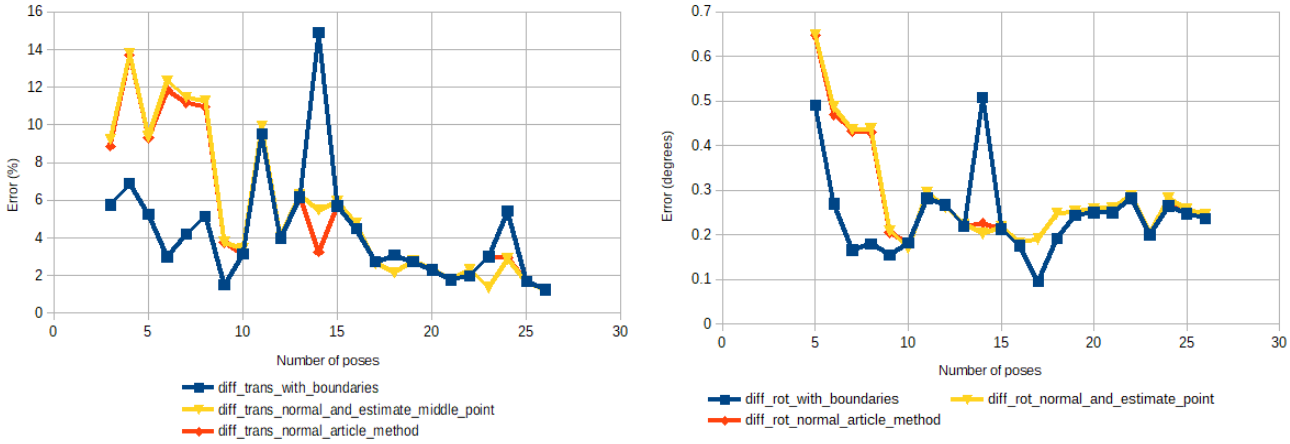


Figure 3.3: Maximal errors in function of a larger number of poses

We then see that from about 15 poses, the results between the 3 methods return fairly similar values. Methods using only normal vectors show similar results, and also seem more robust against possible outliers than the article method (linked for example to poor checkerboard border detection). However, the article method remains more effective in obtaining an accurate estimate of the homothety matrix from a small number of poses. It can be seen that at the end, the maximum difference between the theoretical translation matrix and that obtained via the article is 1.25% (corresponding to about 3mm); such a difference corresponds to the noise included in the simulation of the data, and can therefore be considered as acceptable.

3.2 Precision test in real situation

The robustness of this algorithm could then be evaluated via a set of data taken in real life situations. Normally, the position of the sensors must be the same in relation to the theoretical value.

Similar results can then be observed with the simulated data; again, the article method allows a much better approximation with a single pose. Here again, it can be seen that the results between the three methods become very similar from 15 poses considered. However, the noise is higher than in the simulation here: a maximum error of 0.7° is present on the rotation angles, while a maximum deviation of 8% (corresponding to 11mm) exists in the translation matrix. Such a deviation may be due either to incorrect calibration of the sensors or to unintentional movement of the sensors on their mounting system (leading to the presence of a small unintentional angle of rotation, or to an unexpected translation).

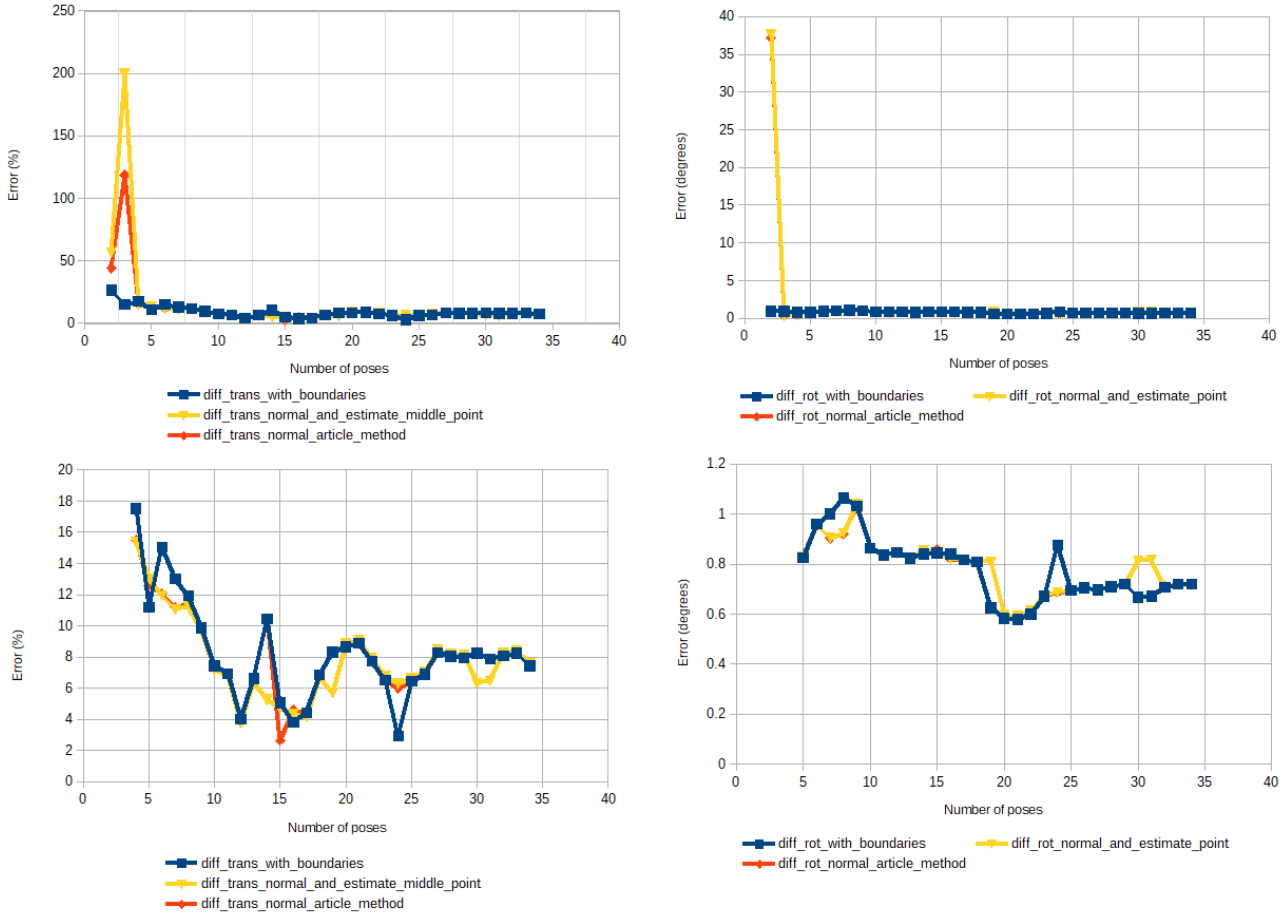


Figure 3.4: Maximal errors with real datas

3.3 Application of the results to datas taken outdoors

A final test was then carried out with the data taken for outdoor use. Unfortunately, the measures taken to search for the homothety matrix between the two sensors proved to be too degraded to allow accurate detection of the checkerboard boundaries. Only the detection of the plan was therefore used.

To obtain the homothety matrix, 24 poses with minimal degradation were used. This number was determined by the results of previous tests, which showed that such a number was more than sufficient to obtain a result similar to the result obtained using the article method.

Even if it is not perfect, it allows to obtain a certain approximation of the homothety matrix, usable for the fusion of the LIDAR and camera data. This approximation also seems consistent with the mounting of the sensors used to take these data. This result could also be confirmed when using data taken on the road: we can see that the fusion of data is consistent with reality.

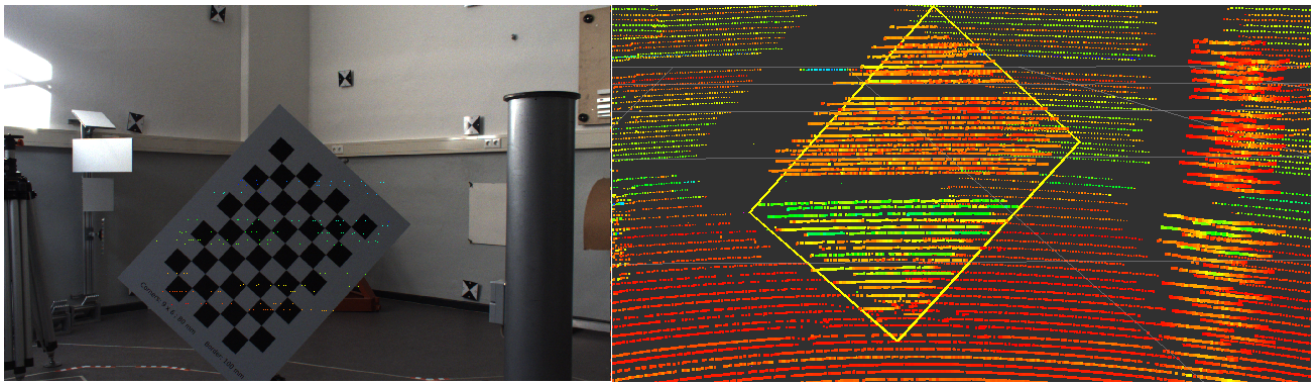


Figure 3.5: LIDAR and camera reprojection

Figure 3.6: Fuse of LIDAR and camera with datas taken on the road

Conclusion

Thanks to tests carried out with several datasets from different sensors, it is now possible to evaluate the methods used in this report for the fusion of sensor data. It can then be seen that the article method is effective in obtaining an accurate estimate of the homothety matrix from a small number of poses; but only if the LIDAR data are not too degraded. In the event that LIDAR data are too degraded, the risk of detecting absurd directions becomes too high to be able to use the article method; it is then better to take as many poses as possible, and to treat only those components of the plan for which the detection is more accurate. In any case, it seems impossible to have a perfectly functional result with a single pose. According to the tests performed, the accuracy of these methods seems to be equivalent from 15 poses. This accuracy seems sufficient for use on embedded systems. The implementation of the method of this article realized during my internship requires however to be tested via other sensors, in order to evaluate its robustness; for example, with other types of cameras, or LIDAR sensors presenting more noise in their measurements. Moreover, this implementation can also be improved, especially in terms of checkerboard border detection. Thus, at the camera level, the implementation of the LSD algorithm must be improved to ensure border detection, in order to be less dependent on the type of checkerboard used for data detection. For LIDAR, better filtering of outliers at borders and detection of border directions is also desirable in order to be able to use all available poses. It could also be interesting to apply interval theory to this implementation, in order to frame errors on the detection of checkerboard components and to perfect the estimation of the homothety matrix. This internship allowed me to perfect my experience on ROS and C++, and to learn how to handle data from different LIDARs and cameras. It also allowed me to work on all the steps of data fusion, and particularly on the temporal and spatial synchronization of data. I was also able to study different methods used for sensor fusion, and evaluate their effectiveness. All this therefore constitutes essential knowledge in my engineering education, which can be applied during my future professional experiences.

Bibliography

[1] Z.Zhang, A Flexible New Technique for Camera Calibration. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(11):1330-1334, 2000.

[2] Atanacio, Gerardo & Gonzalez-Barbosa, José-Joel & Hurtado-Ramos, Juan & Ornelas-Rodriguez, Francisco & Jiménez-Hernández, Hugo & Garcia-Ramirez, Teresa & Gonzalez-Barbosa, Ricardo. LiDAR Velodyne HDL-64e calibration using pattern planes. International Journal of Advanced Robotic Systems. 8. 10.5772/50900, 2011

[3] Lipu Zhou and Zimo Li and Michael Kaess, Automatic Extrinsic Calibration of a Camera and a 3D LiDAR using Line and Plane Correspondences, IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, IROS, October 2018

[4] Gioi, Rafael & Jakubowicz, Jeremie & Morel, Jean-Michel & Randall, Gregory. LSD: A Fast Line Segment Detector with a False Detection Control. IEEE transactions on pattern analysis and machine intelligence. 32. 722-32. 10.1109/TPAMI.2008.300, 2010

[5] Konstantinos G. Derpanis, Overview of the RANSAC Algorithm, May 2010

[6] Ananth Ranganathan, The Levenberg-Marquardt Algorithm, June 2004

[7] FLIR, Grasshopper 3 USB3 Vision Datasheet, 2017

Appendix

Appendix A : Node graphs used

Figure 3.7: Node graph for the fuse of sensor data

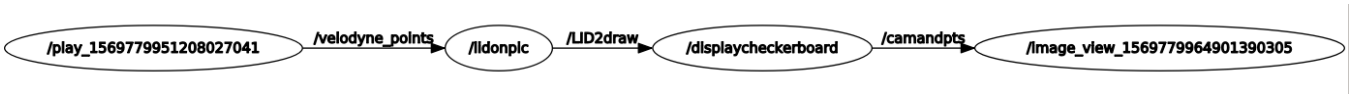
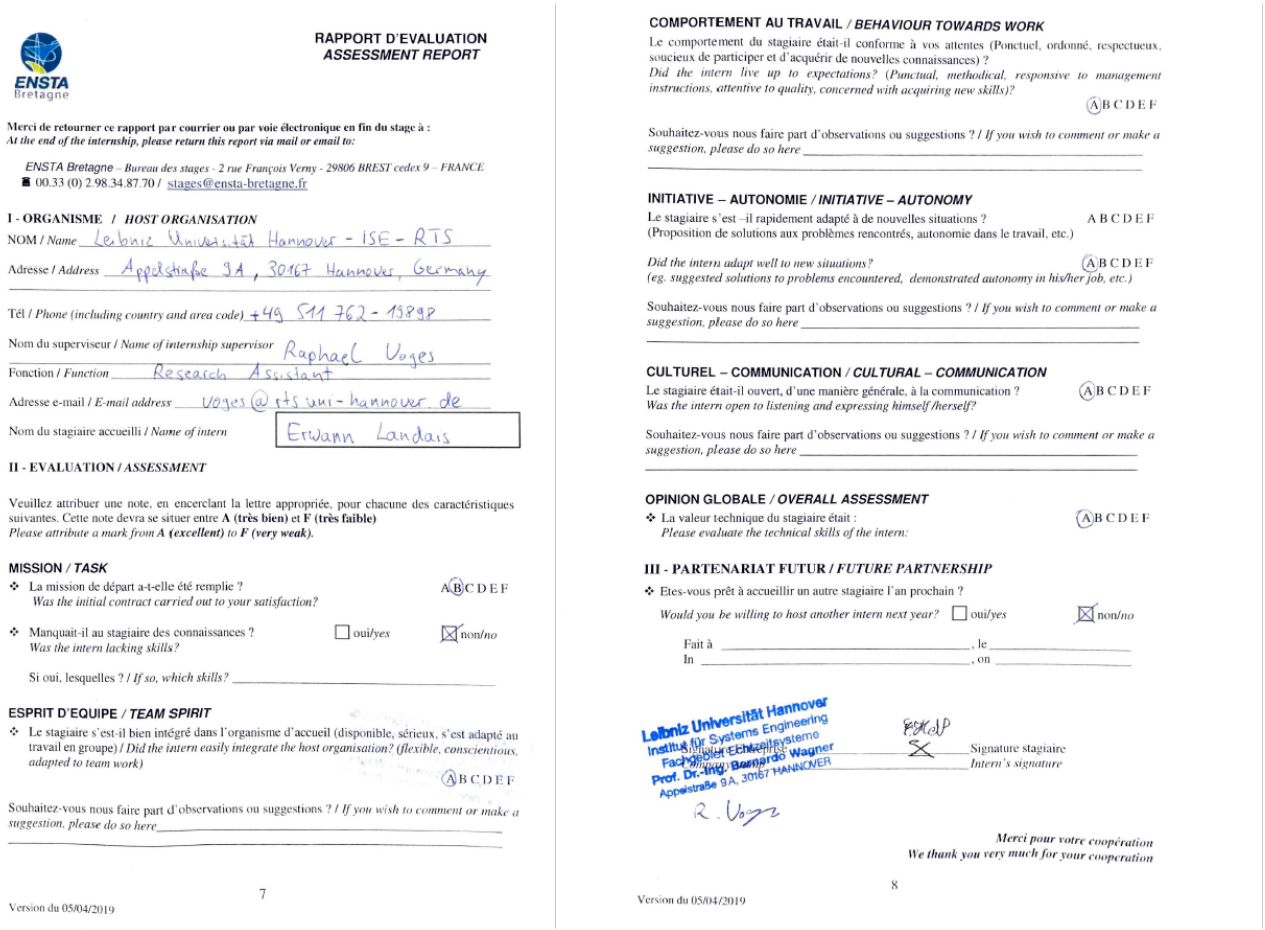


Figure 3.8: Node graph for the evaluation of the results (while listening to a .bag)

Appendix B : Evaluation report



RAPPORT D'EVALUATION
ASSESSMENT REPORT

ENSTA Bretagne

Merci de retourner ce rapport par courrier ou par voie électronique en fin du stage à :
At the end of the internship, please return this report via mail or email to:

ENSTA Bretagne – Bureau des stages - 2 rue François Verny - 29806 BREST cedex 9 – FRANCE
00.33 (0) 2.98.34.87.70 / stages@ensta-bretagne.fr

I - ORGANISME / *HOST ORGANISATION*

NOM / *Name* Leibniz Universität Hannover - ISE - RTS

Adresse / *Address* Appelstraße 9A, 30167 Hannover, Germany

Tél / *Phone (including country and area code)* +49 511 762 - 19898

Nom du superviseur / *Name of internship supervisor* Raphael Voges

Fonction / *Function* Research Assistant

Adresse e-mail / *E-mail address* voges@rts.uni-hannover.de

Nom du stagiaire accueilli / *Name of intern* Erwann Landais

II - EVALUATION / *ASSESSMENT*

Veuillez attribuer une note, en encerclant la lettre appropriée, pour chacune des caractéristiques suivantes. Cette note devra se situer entre **A (très bien)** et **F (très faible)**
*Please attribute a mark from **A (excellent)** to **F (very weak)**.*

MISSION / *TASK*

❖ La mission de départ a-t-elle été remplie ? *Was the initial contract carried out to your satisfaction?* A (B) C D E F

❖ Manquait-il au stagiaire des connaissances ? *Was the intern lacking skills?* ☐ oui/*yes* ☒ non/*no*

Si oui, lesquelles ? / *If so, which skills?* _____

ESPRIT D'EQUIPE / *TEAM SPIRIT*

❖ Le stagiaire s'est-il bien intégré dans l'organisme d'accueil (disponible, sérieux, s'est adapté au travail en groupe) / *Did the intern easily integrate the host organisation? (flexible, conscientious, adapted to team work)* (A) B C D E F

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

7

Version du 05/04/2019

COMPORTEMENT AU TRAVAIL / *BEHAVIOUR TOWARDS WORK*

Le comportement du stagiaire était-il conforme à vos attentes (Ponctuel, ordonné, respectueux, soucieux de participer et d'acquérir de nouvelles connaissances) ?
Did the intern live up to expectations? (Punctual, methodical, responsive to management instructions, attentive to quality, concerned with acquiring new skills)? (A) B C D E F

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

INITIATIVE – AUTONOMIE / *INITIATIVE – AUTONOMY*

Le stagiaire s'est –il rapidement adapté à de nouvelles situations ? A B C D E F
(Proposition de solutions aux problèmes rencontrés, autonomie dans le travail, etc.)

Did the intern adapt well to new situations? (A) B C D E F
(eg. suggested solutions to problems encountered, demonstrated autonomy in his/her job, etc.)

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

CULTUREL – COMMUNICATION / *CULTURAL – COMMUNICATION*

Le stagiaire était-il ouvert, d'une manière générale, à la communication ? (A) B C D E F
Was the intern open to listening and expressing himself /herself?

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

OPINION GLOBALE / *OVERALL ASSESSMENT*

❖ La valeur technique du stagiaire était : (A) B C D E F
Please evaluate the technical skills of the intern:

III - PARTENARIAT FUTUR / *FUTURE PARTNERSHIP*

❖ Etes-vous prêt à accueillir un autre stagiaire l'an prochain ?
Would you be willing to host another intern next year? ☐ oui/*yes* ☒ non/*no*

Fait à _____, le _____
In _____, on _____

Leibniz Universität Hannover
Institut für Systems Engineering
Fachgebiet Echtzeitsysteme
Prof. Dr.-Ing. Bernardo Wagner
Appelstraße 9A, 30167 HANNOVER

Signature stagiaire
Intern's signature

R. Voges

Merci pour votre coopération
We thank you very much for your cooperation

8

Version du 05/04/2019

Figure 3.9: Evaluation Report