Online Cross-Calibration of Camera and LIDAR

Bianca-Cerasela-Zelia Blaga, Sergiu Nedevschi
Computer Science Department
Technical University of Cluj-Napoca,
Cluj-Napoca, Romania

Abstract — **In an autonomous driving system, drift can affect the sensor's position, introducing errors in the extrinsic calibration. For this reason, we have developed a method which continuously monitors two sensors, camera, and LIDAR with 16 beams, and adjusts the value of their cross-calibration. Our algorithm, starting from correct values of the extrinsic cross-calibration parameters, can detect small sensor drift during vehicle driving, by overlapping the edges from the LIDAR over the edges from the image. The novelty of our method is that in order to obtain edges, we create a range image and filter the data from the 3D point cloud, and we use distance transform on 2D images to find edges. Another improvement we bring is applying motion correction on laser scanner data to remove distortions that appear during vehicle motion. An optimization problem on the 6 calibration parameters is defined, from which we are able to obtain the best value of the cross-calibration, and readjust it automatically. Our system performs successfully in real time, in a wide variety of scenarios, and is not affected by the speed of the car.**

Keywords— cross-calibration, camera, LIDAR, laser range finder, edge detection, autonomous driving.

I. INTRODUCTION

Today, autonomous vehicles use a large variety of sensors to navigate in an environment. We want to enhance 2D images with the depth information provided by laser range finders, and add color data taken from cameras to 3D point clouds. Sensor fusion can form the foundation of developing further complex functionalities such as pedestrian detection, traffic signs recognition, and localization and mapping. For this, we need to know how the sensors relate to each other and the world. There are two types of calibrations: intrinsic and extrinsic. The intrinsic parameters of a camera are the focal length, optical center, skew coefficient, and distortion coefficient, while the internal parameters of a LIDAR are the elevation angles of the laser scanners, azimuth angle, and reflectivity. Factory settings can be used for mono-cameras, and LIDARs, but sometimes corrections need to be applied to laser range finders. [1] and [2] offer detailed explanations of the internal structure of a laser scanner – more specifically Velodyne HDL-64E, showing how the laser beams are situated, and employing a least-squares method for initial estimation and refinement.

The extrinsic calibration parameters give the orientation between the sensor and the object we take the measurements from. The location of cameras and laser scanners in the world can be determined if we know the 6 DOF transformations

Figure 1. An example of miss-calibration detection and correction

from the coordinate system of the specific sensor to the 3D world coordinate system. The extrinsic matrix is composed of a 3x3 rotation matrix and a 3-dimensional translation vector. In data fusion, alignment between sensors' rotation and translation represents a problem to be solved. As Lili Huang states in [3], three approaches for LIDAR and camera calibration are: Visible Beam Calibration – observing the LIDAR beams or reflected points (needs infrared cameras), 3D LIDAR based calibration – uses corners or edges of specific calibration objects to obtain a relationship between the two types of representations, and 2D-planar based calibration – observing a plane of an object and solving distance constraints from the camera and LIDAR. A method that falls in the third category is [4], which consists in using a checkboard observed by both sensors at the same time, in different positions and orientations, and estimating the coefficients using a least-squares method.

Many methods use a calibration target seen by both sensors at the same time to find the cross-calibration: [5] uses a black circle as pattern, solving the alignment between the normals of the planes obtained from the detected ellipse in the 2D image and by the laser scanner; in[6], a vertical wall and a checkerboard pattern are used to solve a linear optimization problem using Levenberg-Marquardt algorithm; [7] uses triangular targets that are detected in both images and 3D point clouds, and the cross-calibration is obtained using the Levenberg-Marquardt optimization algorithm, [8] utilizes a dark circle with empty interior, exploiting the reflectivity information from laser scanners; in [9] a checkerboard pattern and planar markers with different geometrical shaped holes are detected in both camera and LIDAR are correlated.

The drawback of most methods is that they use a specific object, that is not available if we want to recalibrate, and they are time-consuming. The most obvious choice of features that appear in both images and 3D point clouds are the edges. In paper [10], the edges are matched using a weighted total

variation norm extracted from the fused depth image. [11] and [12] present different methods to detect sharp features in a point cloud based on Gauss map clustering, which can be further used to align the camera and the LIDAR.

The problem of most calibration methods is that they cannot be applied for real time computer vision. In autonomous driving, having a fixed calibration is not a safe solution, as sensor drift can occur. Due to the interaction with the environment, the camera or the LIDAR can change their positions, making the cross-calibration improper. This can affect the accurate functioning of the system. To correct this, recalibration is needed, but it is not feasible, as it is time-consuming and needs a specific environment. Therefore, it appears the need for online calibration, which can monitor and refine the extrinsic parameters during normal vehicle functioning conditions. In paper [13], the road, which is a flat surface, and an obstacle seen by both sensors are used to cross-calibrate the rotation between a stereo-camera and a laser scanner using MSAC to find the road plane and solving alignment between the point clouds obtained from both sensors. In [14], a fully automatic method for extrinsic calibration based on SLAM is presented. The system is composed of multi cameras fixed on a rotating rig, with little to no overlapping FOV, and requires no patterns. [15] presents a method to automatically calibrate the extrinsic parameters of a camera by using epipolar geometry. The epipoles are used for a mono camera in a similar way they are used in stereo: by matching feature points from 2 consecutive shots. In [16], the goal is to detect in real-time the miss-calibration between camera and LIDAR by superimposing edges from images and laser scanner. [17] uses the Renyi Quadratic Entropy to estimate the position of the sensors related to the vehicle frame.

Our aim in this paper is to present an online cross-calibration system, that is able to correct miss-calibrations during normal functioning of a vehicle. The rest of the paper is structured as follows: Section II presents the overall system design. Sections III, IV and V present the edge detection algorithm for camera and LIDAR and the online cross-calibration method. Section VII presents how the extrinsic LIDAR calibration can be obtained. The last two sections are dedicated to experimental results and conclusions.

II. System Design For Online Cross-Calibration

Figure 2 illustrates the flow of the online cross-calibration algorithm. We first start with known cross-calibration parameters obtained from an offline method such as [4] or [18]. Next, as long as we have frames, we repeat a series of steps. First, detect edges from the 2D images using distance transform. Second, apply motion correction on LIDAR data to correct the representation of 3D points and eliminate errors caused by the speed of the car. Afterward, obtain a range image from the laser scanner, that will be further used to detect edges from the LIDAR data. The last step is to apply the online calibration algorithm – vary the 6 extrinsic parameters, translation on the 3 axes and rotation around x, y, and z axes. If we obtain a better value of the cost function, we update the cross-calibration parameters, if not, we continue with the same values, and repeat the steps.

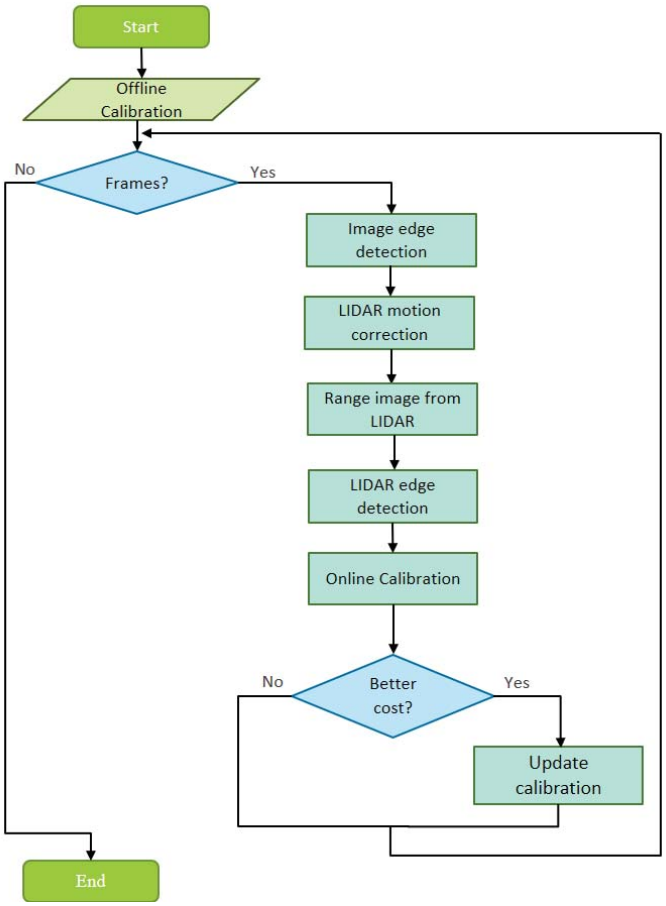


Figure 2. Design of the Online Cross-Calibration system

Temporal synchronization between the two sensors is achieved by taking into reference the timestamps that the measurements get from the camera, the laser scanner, and the GPS. We use interpolation to obtain the exact image frame for the current point cloud, considering the time differences between consecutive frames.

The offline cross-calibration between laser scanner and camera is found by placing a checkerboard calibration pattern in front of the camera and laser scanner in multiple positions and orientations. We can find the normal of the plane pattern in the images using Bouguet's "Camera Calibration Toolbox"[19], by selecting multiple corner points from the checkerboard plane. To find the normal in the LIDAR dataset, we apply an RANSAC plane fitting procedure. These are used to solve a least square minimization problem to find the cross-calibration parameters, which are further refined using the Levenberg-Marquardt method. Details about this procedure can be found in [4].

What we bring new to the online cross-calibration, compared to existing methods, is the motion correction method. To understand this concept visually, think of a point cloud as being circular. In most algorithms or methods that use laser scanner data, all the points from a full point cloud

are considered to have been taken at the same moment, which is not true. In applications that require high accuracy, such as the problem of calibration, this preconception introduces important errors that render the system incapable of functioning. By applying a motion correction algorithm, the time stamp of each point is taken into consideration to adjust the position of the corresponding point, shifting points that have been taken at a later time forward. Thus, the dataset will no longer be a circle, but rather a spiral, with points from the beginning to the end having the same timestamp. We use the VICP algorithm [20] to correct the distortions from the laser scanner measurements, that appear due to the motion of the car or objects moving in the scene.

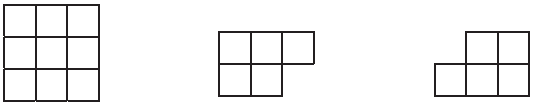
In our method, we have two systems working together: the offline cross-calibration and the online cross-calibration. The first one is mandatory to find the initial calibration parameters, when the sensors are first placed on the car, and it is done only once. These initial values are used as input to the second system, which will run at all times during car driving since sensors can drift during the interaction with the surrounding environment. It is important to note that if the sensors' position on the vehicle changes, we have to recalibrate using the offline method first, then monitor the calibration with the online method. We focus on the online cross-calibration system, with details about its implementation being presented in the next sections.

III. CAMERA EDGE DETECTION

In this section, we focus on the edge detection algorithm for 2D images, which is the distance transform. The result of the transform is a grayscale image that looks similar to the input image, except that the intensities of points are changed to show the distance to the closest boundary from each point[21].

We first obtain a grayscale image, in which each pixel takes the maximum value between the difference between it and its 8 neighbors. Next, we used the Chamfer based distance transform method, which requires scanning the binary image from two sides, making the computation very fast. The main steps of the algorithm are:

1. Choose a 3x3 mask which is further decomposed into two parts:



2. A double scan (first top-down, left-right and second bottom-up, right-left) of the image (Figure 3**)**, using the previous masks, is required to update the distance on the DT image.

   First perform an initialization step, in which we compute the edges by updating each pixel to the maximum value between the difference between the current pixel and its 8 neighbors. Then we apply the update:

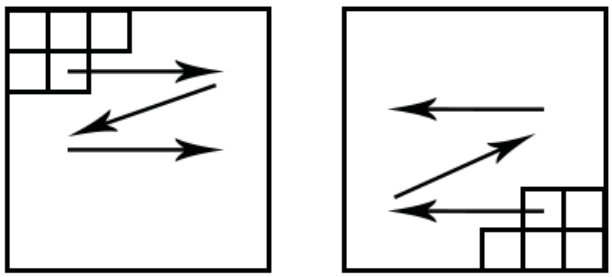


Figure 3. Double scan using the decomposed mask

$$DT(i,j) = \min_{(k,l)\in Mask}(DT(i+k, j+l) + weight(k,l)) \quad (1)$$

where $weight(k,l)$ is *wHV* if the direction *(k,l)* is horizontal or vertical from the center of the mask, or *wD* if the direction is diagonal. The relationship between the 2 values should be $wD = wHV \cdot \sqrt{2}$; in our implementation, we use the values *wD*= 7 and *wHV*= 5, because they give better visual results, as it can be seen in Figure 4.

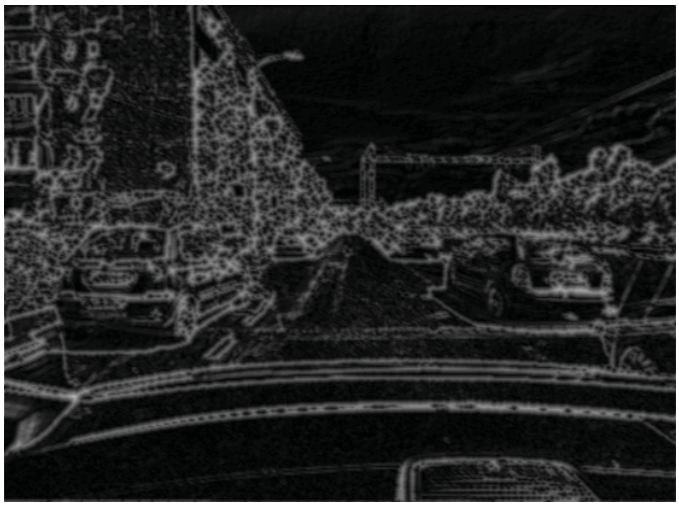


Figure 4. Distance Transform image result

IV. LIDAR EDGE DETECTION

For detecting edges in 3D point clouds obtained from LIDAR readings, we follow two steps. The first step is to obtain the range image, which is a 2D representation of the laser scanner points with depth information. We use [22] to obtain the range image, denoted with R, which contains information about the point coordinates and the range data.

On the range image, we apply the second step, which is a filtering of points based on depth discontinuities. We denote a point from LIDAR with L, and its left and right neighbors with L_{left}, respectively L_{right}. We obtain the edges by taking into consideration the depth of the points and the relationship between neighbors as follows. The difference between neighbors is the maximum value of their depth discontinuity:

$$m = \max(L_{left} - L, L_{right} - L, 0)^{\gamma} \quad (2)$$

where $\gamma = 0.5$. Next, we filter after the depth of each point:

$$E.range = \begin{cases} -\infty, if\ R < 8\ and\ \log(R^{0.26}) > m \\ -\infty, if\ \log(R^{0.5}) > m \\ m, otherwise \end{cases} \quad (3)$$

Where E is the new point cloud, composed of points that are situated on edges in the original 3D data. The values of 0.26 and 0.5 were obtained after measurements, and are specific for the sensors we use in our system. The filtering after depth in 3D data is needed because, if we take into consideration only the neighbors, there would be a high density of edges at big distances, but the information from images is not detailed enough to be relevant.

V. Online Calibration

Given a cross-calibration matrix, we can project the laser points from the edge point cloud to the edge image, using the principles of pinhole camera projection. These projections will be used to infer a cost function which exploits the fact that we have a correct calibration if the edges from the LIDAR are correctly superimposed over edges from the image. We take the cost function as being the sum of the product between the information from the edges over the last *w* frames.

$$C = \sum_{j=f-w}^{f} \sum_{i=1}^{N} \sqrt{DT_i \cdot E_i.range} \quad (4)$$

Where f denotes the current frame number, w is the number of frames used (window size), N is the number of projected LIDAR points on the 2D image, DT is the edge image, and E is the edge point cloud, from which we take only the current point with the range information.

In order to determine the best calibration, we vary the 6 parameters in the following way: we decrease them with a value, we let them constant, or we increase them with a value. Therefore, we will compute 729 values of the cost function, from which we will choose the maximum as being the best value of the calibration. The cost function is guaranteed to converge only if there is a small perturbation of the 6 parameters, depending on the density of data provided by the laser scanner. When we obtain a better value for the cost, we update the calibration.

The cost function will be optimized only if we start from correct cross-calibration parameter values, otherwise, it will never reach the optimum value. Therefore, we need a way to know when to stop updating if the new values only worsen the cross-calibration. A method would be to observe how the cost function changes when the 6 parameters are varied, for all possible variations. We use the miss-calibration detection from [16], which builds a Gaussian probability distribution over the varied values of the calibrations, and updates the 6 cross-calibration parameters only when the probability of being correct is higher than a given threshold. This ensures that, when the two sensors are severely miss-aligned, the values of the calibration parameters will not be updated.

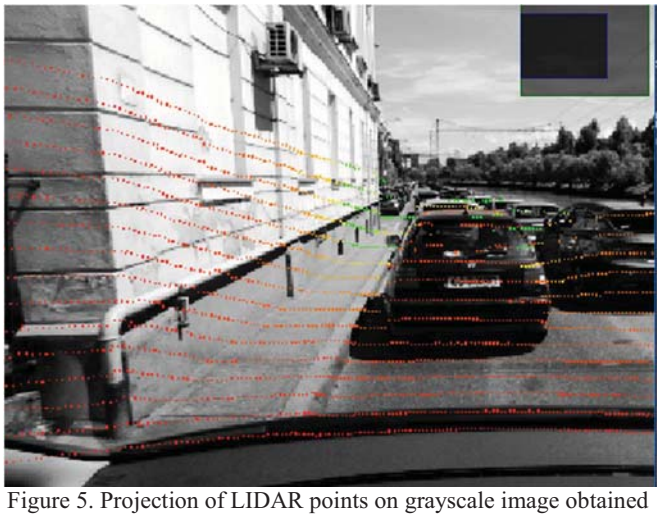


Figure 5. Projection of LIDAR points on grayscale image obtained from the camera

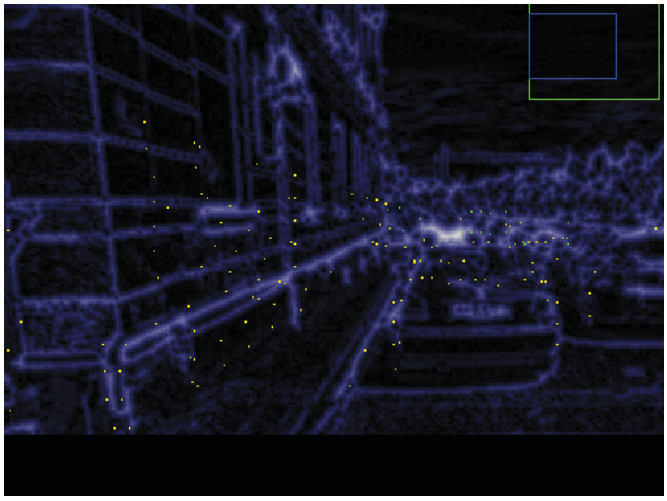


Figure 6. Edges from LIDAR data projected on edges obtained applying distance transform

VI. LIDAR Extrinsic Calibration

In our system, the calibration parameters can be generally denoted as $\theta_s^r = \{x_s^r, y_s^r, z_s^r, \alpha_s^r, \beta_s^r, \gamma_s^r\}$, where s represents the sensor, camera or LIDAR, and r is the reference coordinate system. The first 3 represent the translation vector, while the last 3 are used to compose the 3x3 rotation matrix.

The camera has an intrinsic calibration matrix, composed of a rotation matrix and a translation vector, denoted:

$$K_C^i = \begin{bmatrix} R_C^i & T_C^i \\ 0\ 0\ 0 & 1 \end{bmatrix} \quad (5)$$

where K_C^i means the transformation from camera coordinates to world coordinates, R_C^i is the rotation matrix, and T_C^i is the translation matrix.

After applying the offline calibration algorithm, we obtain an extrinsic calibration matrix, which transforms points from camera coordinates to world coordinates, and is denoted as:

$$K_C^W = \begin{bmatrix} R_C^W & T_C^W \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \tag{6}$$

After applying the cross-calibration, we obtain the transformation between laser scanner and camera, transferring the points from the LIDAR coordinate system in the coordinate system of the camera.

$$K_L^C = \begin{bmatrix} R_L^C & T_L^C \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \tag{7}$$

We can obtain the extrinsic calibration matrix of the LIDAR relative to the world coordinate system:

$$K_L^W = K_C^W \cdot K_L^C \tag{8}$$

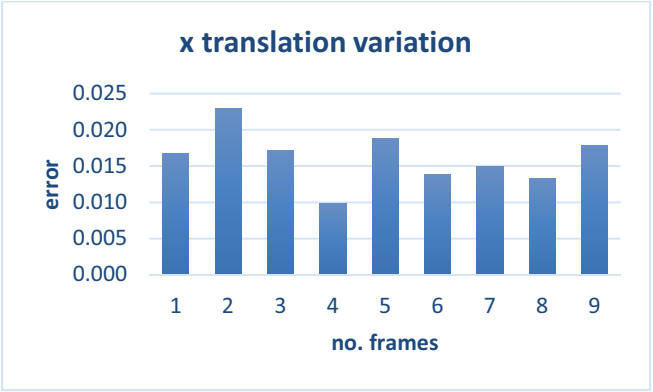
Therefore, if we know the extrinsic matrix of the camera relative to the world, and the extrinsic cross-calibration between the laser scanner and the camera, we can find the extrinsic calibration between the LIDAR and the world.

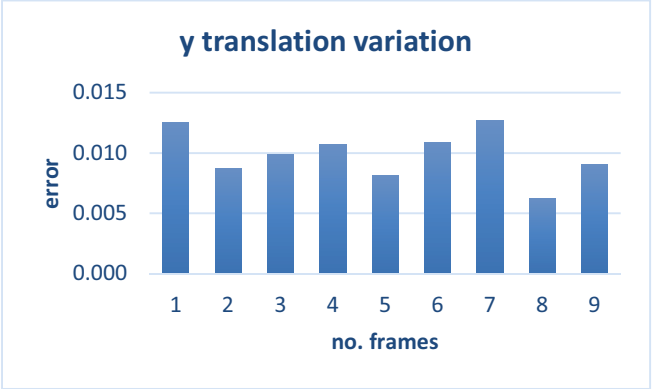
VII. Experimental Results

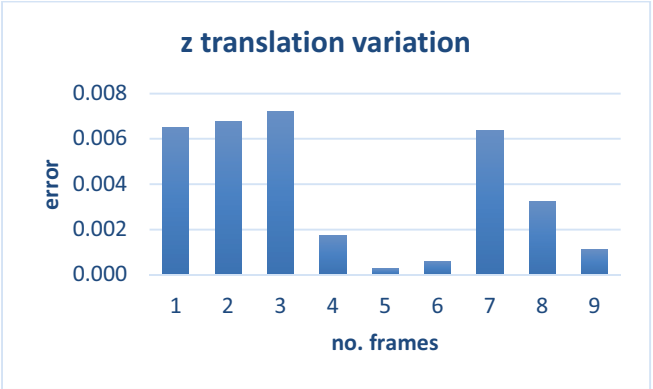
Our system is developed in C++, using the following libraries: PCL (Point Cloud Library), OpenCV, Boost, Qt, and Eigen. The hardware on which we ran the tests is Intel x64 processor, with the frequency of 4 GHz, and 64 GB RAM memory. The operating system is Windows 10.

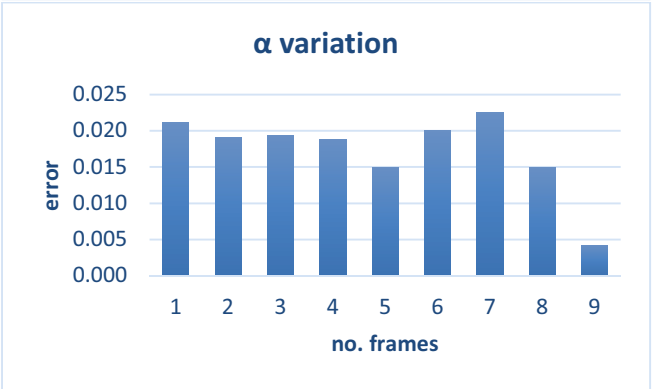
As sensors, we use a Velodyne VLP-16 sensor with 16 layers of data oriented between -15 and +15 degrees vertically, covering a vertical field of view of 30 degrees. The sensor covers a 360-degree horizontal FOV, while spinning at 10 Hz, and provides 300.000 points per second. It can cover a distance up to 100m, with an error of -/+3cm. The camera is JAI BM 141GE, that gives monochrome images. The coordinate systems of the world and the two sensors have the z-axis pointing forward, the x-axis – to the right and the y-axis – to the ground.

We performed experiments to determine the speed and accuracy of our algorithm. The algorithm running time is between 16ms for one frame and 90ms for 9 frames, the most time-consuming part being the grid search step. To track the miss-calibration correction, we have randomly altered the 6 extrinsic calibration parameters with noise, and compared the obtained values with the ground truth cross-calibration, which is known from the offline algorithm. Each translation parameter was modified with values between 1 and 2cm, and each rotation value was altered with up to 2 degrees, which are quite noticeable. Higher offsets would lead to having a low probability that the calibration is correct, which is computed in the miss-calibration step, therefore they would not be updated and the car should be stopped. The experiments were performed on a 1.900 frame log, in which we have variation in the speed of the car, pedestrians moving, and cars coming from the opposite direction. As metrics, we use the Mean Squared Error (MSE) between the values of the cross-calibration obtained from the offline method, which is considered the ground truth, and the values obtained from the online cross-calibration method.



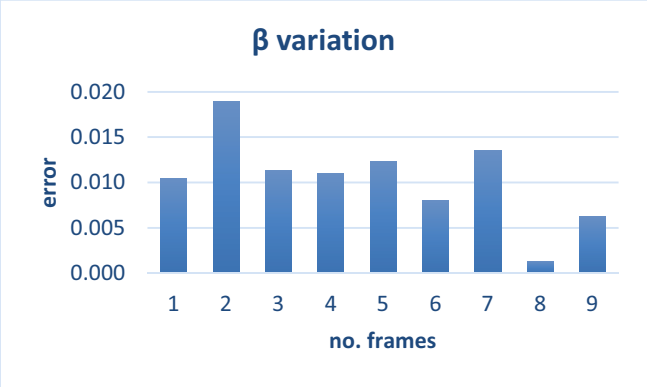


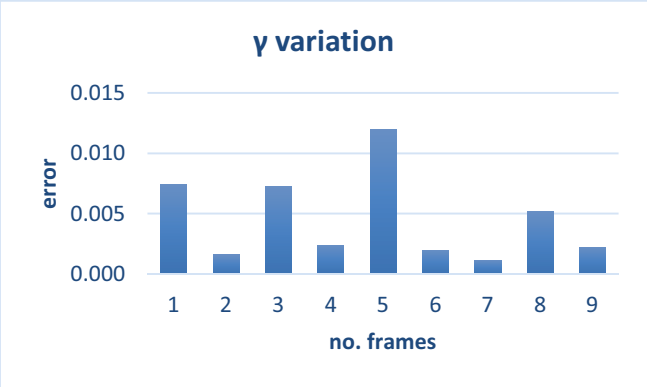




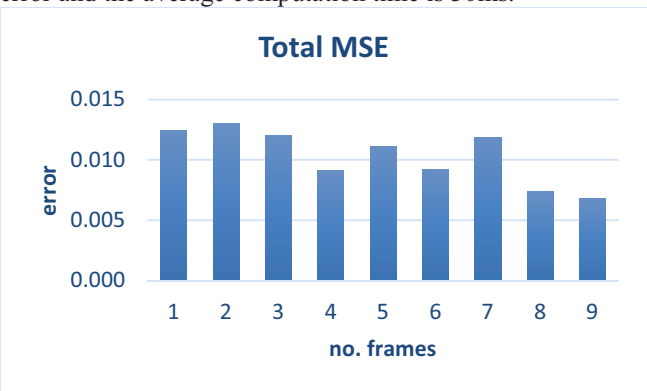
From the plots with the variations of the 3 translation parameters and 3 rotation parameters, we can observe the errors we obtain for each of the 6 cross-calibration parameters when we vary the number of frames. The biggest variations can be observed for the translation on the z-axis and for the γ angle, from which we can notice where we obtain high or low

errors, depending on the window size. This was used to decide on the optimal number of frames to be applied to the online cross-calibration algorithm, in such a way that the error between the ground truth and the refined result is minimized.





The error decreases when using more frames, the lowest error of 0.007 being obtained when having a window of 9 frames, as seen in the plot of MSE. The only downside is the computation time, which is quite high for such an application. Therefore, we choose to use 4 as the number of frames for the online cross-calibration algorithm, because it gives a low error and the average computation time is 30ms.



In Figure 7, the 3D points obtained from the laser scanner are projected onto the 2D image, with noise added to the cross-calibration between the camera and the LIDAR. After applying our online cross-calibration algorithm, we can successfully refine the calibration parameters and correct them; the result can be seen in Figure 8.

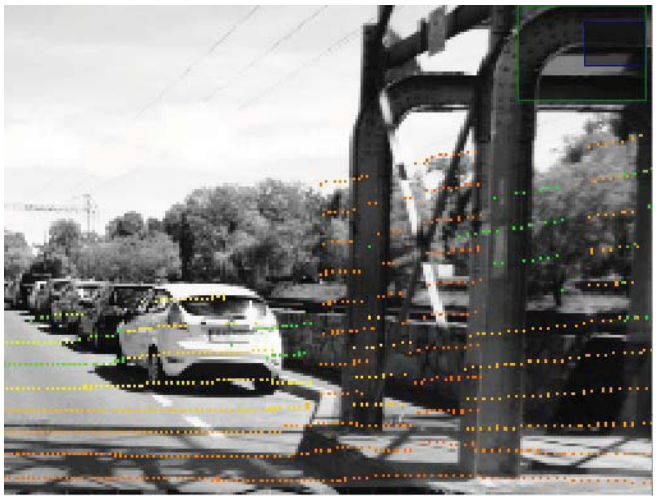


Figure 7. Miss-calibrated projection of LIDAR data on image

Figure 8. Corrected cross-calibration and projection of LIDAR data on image

In Table 1, from the results obtained in the miss-calibration tracking, it can be seen that our implementation has a higher degree of accuracy than other existing methods. We don't obtain an improvement when refining the translation on the y-axis, which is vertical because the laser scanner we use has only 16 layers; this gives a low density of points on the vertical edges, compared to the information obtained from edges in an image, which is dense.

Table 1. Online cross-calibration results

	Translation (m)			Rotation (degrees)		
Paper	x	y	z	α	β	γ
[23]	0.0045	0.0052	0.0046	0.38	0.39	0.44
[16]	0.02	0.014	0.006	0.672	0.628	0.476
[6]	0.305	-0.005	-0.426	-0.15	0	0.27
Our system	0.002	0.015	-0.005	-0.016	0.002	0.01

The cross-calibration algorithm can also be used to refine the results of an offline calibration algorithm, even when the sensors' position doesn't change, being able to correct small mistakes. In Figure 9, we have plotted the errors between the online and offline cross-calibration methods, when no noise has been added to the sensors' readings. The offline method has high errors for two rotation parameters, which are corrected with the online method.

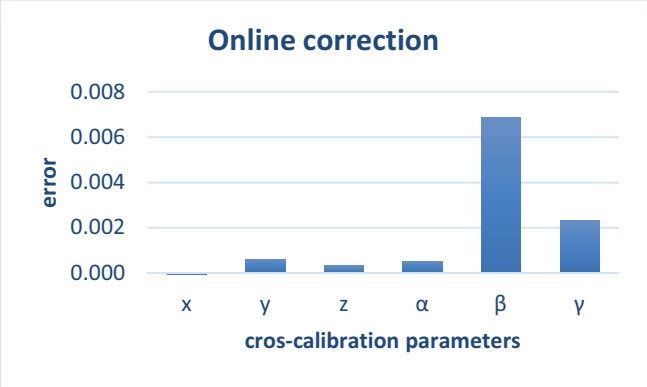


Figure 9. Offline cross-calibration correction

VIII. Conclusions

This work presents a new online solution for the cross-calibration of camera and LIDAR. Our system can detect miss-calibrations if sensor drift occurs during vehicle functioning. We are able to correct the cross-calibration between the camera and LIDAR, and also extract the extrinsic calibration parameters of LIDAR. We have obtained better results than previous existing cross-calibration methods, as seen in section VII because the most important improvement we bring is the usage of a motion correction algorithm, which eliminates distortions from the laser scanner that come from vehicle motion or objects moving in the scene. We also create a range image that is an organized representation of the 3D point cloud. By fusing the two sensors, camera, and LIDAR, we obtain a 2D image containing range information. The experimental results demonstrate that our solution is able to run in real time, which makes our online cross-calibration algorithm a convenient component of a driving assistance solution.

Acknowledgment

This work was supported by the MULTISPECT grant (Multispectral environment perception by fusion of 2D and 3D sensorial data from the visible and infrared spectrum) of UEFISCDI, project code PN-III-P4-ID-PCE-2016-0727, contract number 60/2017.

This work was also supported by the EU H2020 project, UP-Drive under grant nr. 688652.

References

[1] F. M. Mirzaei, D. G. Kottas, and S. I. Roumeliotis, "3D LIDAR-camera intrinsic and extrinsic calibration: Identifiability and analytical least-squares-based initialization," *Int. J. Rob. Res.,* vol. 31, pp. 452-467, 2012.

[2] C. Glennie and D. D. Lichti, "Static Calibration and Analysis of the Velodyne HDL-64E S2 for High Accuracy Mobile Scanning," *Remote Sensing,* vol. 2, p. 1610, 2010.

[3] L. Huang, "Lidar, camera and inertial sensors based navigation techniques for advanced intelligent transportation systems," University of California, Riverside, 2010.

[4] L. Huang and M. Barth, "A novel multi-planar LIDAR and computer vision calibration procedure using 2D patterns for automated navigation," in *2009 IEEE Intelligent Vehicles Symposium*, 2009, pp. 117-122.

[5] H. Alismail, L. D. Baker, and B. Browning, "Automatic Calibration of a Range Sensor and Camera System," in *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, 2012, pp. 286-292.

[6] G. Pandey, J. McBride, S. Savarese, and R. Eustice, "Extrinsic Calibration of a 3D Laser Scanner and an Omnidirectional Camera," presented at the 7th IFAC Symposium on Intelligent Autonomous Vehicles, IAV 2010 - Proceedings, 2010.

[7] S. Debattisti, L. Mazzei, and M. Panciroli, "Automated extrinsic laser and camera inter-calibration using triangular targets," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 696-701.

[8] S. A. R. F, V. Fremont, and P. Bonnifait, "Extrinsic calibration between a multi-layer lidar and a camera," in *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 2008, pp. 214-219.

[9] M. Velas, M. Spanel, Z. Materna, and A. Herout, "Calibration of RGB camera With Velodyne LiDAR," *WSCG 2014 Communication Papers Proceedings,* pp. 135-144, 2014.

[10] J. Castorena, U. S. Kamilov, and P. T. Boufounos, "Autocalibration of lidar and optical cameras via edge alignment," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 2862-2866.

[11] S. Sim, J. Sock, and K. Kwak, "Indirect Correspondence-Based Robust Extrinsic Calibration of LiDAR and Camera," *Sensors,* vol. 16, p. 933, 2016.

[12] C. Weber, S. Hahmann, and H. Hagen, "Sharp Feature Detection in Point Clouds," presented at the Proceedings of the 2010 Shape Modeling International Conference, 2010.

[13] C. H. Rodríguez-Garavito, A. Ponz, F. García, D. Martín, A. d. l. Escalera, and J. M. Armingol, "Automatic laser and camera extrinsic calibration for data fusion using road plane," in *17th International Conference on Information Fusion (FUSION)*, 2014, pp. 1-6.

[14] G. Carrera, A. Angeli, and A. J. Davison, "SLAM-based automatic extrinsic calibration of a multi-camera rig," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2652-2659.

[15] M. Miksch, B. Yang, and K. Zimmermann, "Automatic extrinsic camera self-calibration based on homography and epipolar geometry," in *2010 IEEE Intelligent Vehicles Symposium*, 2010, pp. 832-839.

[16] J. Levinson and S. Thrun, "Automatic Online Calibration of Cameras and Lasers," *International Symposium on Experimental Robotics (ISER),* 2013.

[17] W. Maddern, A. Harrison, and P. Newman, "Lost in translation (and rotation): Rapid extrinsic calibration for 2D and 3D LIDARs," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 3096-3102.

[18] G. Atanacio-Jiménez, J.-J. González-Barbosa, J. B. Hurtado-Ramos, F. J. Ornelas-Rodríguez, H. Jiménez-Hernández, T. García-Ramirez, *et al.*, "LIDAR Velodyne HDL-64E Calibration Using Pattern Planes," *International Journal of Advanced Robotic Systems,* vol. 8, p. 59, 2011.

[19] J.-Y. Bouguet, "Camera Calibration Toolbox for Matlab," 2003.

[20] S. Hong, H. Ko, and J. Kim, "VICP: Velocity updating iterative closest point algorithm," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 1893-1898.

[21] S. Nedevschi. *Distance Transform (DT). Pattern Matching using DT.* Available: http://users.utcluj.ro/~nedevschi/PR/labs/prs_lab_04e.pdf

[22] D. Holz, A. E. Ichim, F. Tombari, R. B. Rusu, and S. Behnke, "Registration with the Point Cloud Library: A Modular Framework for Aligning in 3-D," *IEEE Robotics & Automation Magazine,* vol. 22, pp. 110-124, 2015.

[23] A. Napier, P. Corke, and P. Newman, "Cross-calibration of push-broom 2D LIDARs and cameras in natural scenes," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 3679-3684.