Vision-Based Lidar Segmentation for Accelerated Scan Matching

Burtin. Gabriel[1], Bonnin Patrick[2], and Malartre Florent[1]
[1] 4D-Virtualiz, Clermont-Ferrand, France
[2] LISV, Velizy, France
Email: gabriel.burtin@4d-virtualiz.com; patrick.bonnin@uvsq.fr; florent.malartre@4d-virtualiz.com

Abstract—A vision-based algorithm brings a fast segmentation process to a 2D lidar point cloud. Extracted features allow us to set up a segment-based scan matcher. This matching is one of the steps for the localization. Features also give semantic information about the environment. The detection of a corner or a door indicates a potential encounter with human beings. Aware of this "danger" area, the robot will be able to adapt its speed and define areas of focus to the vision algorithms. Indeed, vision is known for its high computation load. The focus provided by the lidar diminishes the area in the image to be analyzed and reduces the load.

Index Terms—2D lidar segmentation, scan matching, safe navigation, sensor fusion, indoor environment

I. Introduction

The robot of tomorrow is built to work in the factory of the future in a cluttered, yet structured and human environment. The robot should navigate between locations, with a possibility of encountering humans. We need a navigation system to be able to work fast, be reliable, and take advantage of the structured characteristic of the environment. This navigation task can be divided into sub-tasks: localization, mapping, path planning, control, and safety management. All of these tasks must be performed within an embedded computation unit on the robot. This means that each task must consume computational resources as little as possible. The robot is moving and needs a regular update on its status. To perform this navigation, the robot has embedded sensors to perceive its environment and match the detected features with a list of features previously stored. A popular approach is the grid-based representation of the environment in which each cell is a probability of the presence or absence of an obstacle [1]. Another approach is the use of beacons, artificially placed beforehand [2], largely used in early versions of the robocup soccer (international competition of robotics, http://www.robocup.org). These approaches do not fit our case, which is navigation in a hospital or a factory. This environment implies that any physical modification is denied. The environment is also large, which means that the grid would be either low resolution or high dimension. The first case brings the risk to be stuck, and the second requires heavy computation. To address these multiple problems, we rely on two basic and well-known sensors: the lidar and the camera. The first one gives fast but partial information about the environment: distances in a horizontal plane section. The second gives high-level information (doors, humans, etc.) but requires more computation time. Using sensor fusion, we aim to guide, by lidar information, the camera algorithm toward the area of interest. In other terms, the processing of lidar data is going to speed up the process by focusing the camera on a region of interest (ROI) inside the frame. These regions can be pedestrian encounter areas and the process, a human detection algorithm. In this paper, we focus on the first stages of this study: feature extraction from lidar data and defining regions of interest to cooperate with the camera. This efficient processing leads to a fast scan matcher which provides basic localization. It also cooperates with other sensors by pointing out elements of interest (corners/doors). Vision algorithms, under development, will include robust embedded real-time constraints [3], [4] taking advantage of multi-threading with multi-core ARM architectures.

Manuscript received December 23, 2017; revised March 20, 2018.
Corresponding author email: author@hostname.org.
doi:10.12720/jcm.13.3.139-144

II. Lidar Data Segmentation

Lidar data is a 1D depth buffer generated by several beams of rays. To extract substantial information, we need to process efficiently the data. Several segmentation methods already exist, and some of them have already been tested and compared [5]. Those methods rely on different aspects of the point cloud or pixels. In one hand, global methods search for features with all points at the same time: Hough or Ransac [6]. Unfortunately, they need an important amount of time, and the result contains too much false positive features detected. On the other hand, local methods, applying sliding window in the list of points, are able to extract the features recursively [7] or iteratively [8], [9], [10]. In image processing, the list of points is provided by an edge linking process. These line finding algorithms are faster and less likely to produce false positive. In the lidar case, distances are sequentially measured and sorted, either clockwise or counterclockwise. The following hypothesis can be formulated: if a ray hits a 3D plane, the following (and preceding) rays are more likely to hit the same surface. Considering the hypothesis and criterion, iterative

methods are the most suited and natural. They spare the edge linking process and benefit the organized status of the 2D points provided by the sensor. A strong iterative segmentation with an implementation based on a double Kalman filter is given by [8]; the use of two Kalman filters to process the entire lidar point-cloud introduces heavy computation complexity and requires a lot of parameters to be tuned. We directed ourselves toward a simpler, yet robust and known method [10].

A. Wall–Danielsson Application

This was originally introduced with vision, particularly to simplify the extracted outline and polygonal shapes from pictures. With only a single parameter, wich allows us to drive the plasticity of segment detection. This method, initially created to link pixels connected with each other, is able to work even with distant points. The main advantages of the Wall–Danielson (WD) are its efficiency, speed, and deterministic behavior (contrary to Ransac). To perform this detection, it compares the difference between a threshold and a ratio. The ratio is between the surface of the polygon formed by all of the points and the distance between the first and last points. The main issue brought by this one-parameter implementation is that a bad threshold has large implications. A severe threshold denies any long segments to be detected: they do not fit the condition; only small segments are extracted because of the sensor noise. A soft threshold is also problematic: not detecting small variations is good, corners take more steps to be detected, and the equation of the wall is biased by the last points of the segment. This issue has been addressed by WD [10], [9]. The autors consider a back-stepping algorithm. A post-process is therefore needed to remove the extra points after the corners. Considering execution time, it is better to get rid of this post-process. Regarding the noise, the algorithm does not take into account its variation. The noise associated with the segment will be computed later using the parameters of the extracted segment. Indeed, the model of the noise, Gaussian distributed, can be transmitted to the equation of the extracted segment.

B. Cascade Filter

The noise has clear consequences: if we choose the wrong threshold, elements such as doors disappear. Because of the small difference in ranges data, an easy threshold will smooth the surface. If doors disappear, we cannot rely on this information to obtain longitudinal localization inside hallways. Depending on the granularity of the elements that we want to extract, we need to adjust the threshold parameter. This forces us either to run the algorithm several times with different parameters or to process again the segments extracted by the very first segmentation. In light of the complexity of each operation, re-using the extracted segments is much faster: from a dataset containing N points, a maximum of N/3 segments is expected. A reduced dataset means less computation, and our objective is precisely to gain time. The decision made was to work directly with the segments. Two operations have been implemented to filter the segment dataset. The first operation is a fusion between segments. This means observing two segments and deciding whether or not they have been split by mistake regarding the criterion given. If they have been noted as "miss-split," i.e., they belong to the same 3D plane, they are fused: the set of 2D points contained in each segment's data is gathered inside one segment, and the parameters of the new segment are computed again. The fusion decision is motivated by a set of rules driven by the geometrical constraint of the environment (Fig. 1). They belong to the same 3D surface (plane); therefore, they are consecutive (i and i + 1), parallel, and aligned (d), and extremities are close (D).

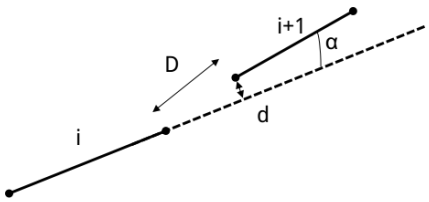

Fig. 1. Fusion criterion

Applying the fusion process to the whole dataset of segments returns a new set with fewer but longer segments. However, if small elements such as a bin or a table foot are laid against the plane, the first condition (consecutive segments) is not complete. In that case, the two sides of the same plane, even if correctly segmented, cannot be fused. Depending on the post-process, the important element is to obtain strong and long segments. Considering the fact that small segments are often too unstable in terms of angle, their presence may induce too much noise in the post-segmentation algorithms. To eliminate the effect of those small segments, a fast erosion filter is used upon the remaining segments. All of these segments are deleted from the dataset using this erosion algorithm. When all of the small blocking segments are removed, a second layer of the fusion algorithm is applied. Depending on the case intended for the segment dataset, the next algorithm can access the data at any level (Fig. 2). Note that the erosion algorithm cannot be applied first as it will remove small segment candidates of the fusion algorithm.

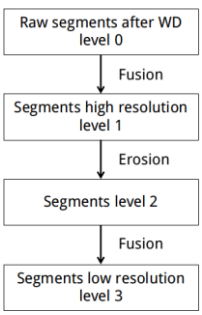

Fig. 2. Cascade filter process.

III. Feature Extraction

Several applications can be found with this segmentation of the lidar data [11], [12]. A faster scan matching can be computed using two successive datasets. Contrary to [13], [14], and [15], in our case, we are able to use a reduced amount of data. All of the data are concentrated inside the remaining segments extracted with high-level segmentation (Fig. 2). Another application is the ability to detect semantic information: the lidar gives us a plane cut of the scene. Using detailed scan (i.e., low-level segmentation), we can detect the shape of common elements (doors).

A. Segment Matching

To perform an efficient scan matching, it is necessary to be able to match points of interest between frames. Thus, with the transformation between each point of interest, we can compute the rigid transformation between the two frames. Segment matching is a complex task: few parameters can describe appropriately the segments. These parameters can be considered only to be two if the representation of the segment is (ρ, θ) [16]. A common similarity weight would only consider those parameters. This leads to many mismatches because a structured environment has several segments with the same parameters (ρ, θ). We chose to use the representation (ρ, θ, COG, length). The COG is the middle of the segment (Center Of Gravity). Keeping the length of the segment is useful in estimating the strength and interest in the segment. A long segment is more likely to be very stable and be seen for a long period by the perception system. Therefore, the new similarity weight considers all of the coefficients. The final goal is to match segments from a set of *n* segments in the previous frame with segments from a set of *m* segments in the current frame. Given the set of extracted segments, we seek every (i, j) couple of related segments. This gives us an *n* by *m* weight matrix. The lower values of the matrix give us the potential matches. Computation of the matrix has to be improved: the computational cost increases as *m* and *n* increase. Two mechanisms are implemented to improve efficiency. The first one is a movement prediction: we use the previous velocity of the robot as an input. When the robot is spinning, long range segments tend to have a high weight because of the large changes in the COG position. Predicting the new position gives better weights and improves significantly the matching between segments. This also allows us to introduce an accelerated weight: we first compute the distance between COGs. If this distance between segments is bigger than a certain threshold, then the weight is given the maximum value without the rest of the computation. The smart weight spares the computation cost of the other elements of the complete weight. The second improvement mechanism is focused on the computation of the weights only on a certain area of the weight matrix. The movement prediction not only allows us to reduce the mean value of the weight, it also gives a hint on the position of potential matches inside the matrix. If the robot goes straight, the matched values are concentrated on the diagonal and around. When the robot spins clockwise, this diagonal tends to translate to the lower triangle. Logically, when the robot spins counterclockwise, the matches are found in the higher triangle of the match matrix. Then, knowing the robot's previous move, we can focus on the most important part of the match matrix. Scan matching accelerated process from 2D lidar data is composed of the following steps:

1) Extract the high-level segments
2) Predict the position of previous segments
3) Compute the weight on the focused area of the matrix
4) Determine every (i; j) couple, i.e.,., matrix minima
5) Compute the rigid transformation using matched segment parameters.

All those mechanisms allow us to accelerate the scan matching with segment-based features.

B. Camera Fusion

Using another level of segmentation can be interesting in detecting high-level or semantic information such as doors and corners. Indeed, a door or a corner means that a person could suddenly appear in front of the robot because of the limited field of view. Failure of the robot to anticipate the possibility of an encounter, depending on the speed and weight of the robot, could lead to severe injury. Having the ability to detect this feature is important to adapt the robot speed and behavior. To perform this sort of detection, we relied on the low-level segmentation because it contains more details about the environment. The corner and the door were described as a set of conditions [8] applied to the set of segments: they allowed us to detect the presence and extract the position of these features. Detecting doors and corners to establish an encounter area is crucial for the safety task. The awareness focuses the heavy computing process in a small area. Once the feature (door or corner) has been identified inside the lidar frame, we need the projection in the image generated by the camera. Equation (1) is the projection of this 3D point in the camera frame. It requires the camera intrinsic parameters (K) and the rigid transformation between the lidar and the camera frame (T). After this projection of the lidar point (P) in the camera image (p), we can determine an ROI and focus. This allows two things: a lower computation cost and a higher chance of finding a pedestrian.

$$p = K * {}^{camera}T_{lidar} * P \quad . \tag{1}$$

IV. Results

A. Set-up

Preliminary tests have been run on a professional virtual platform, a realistic and advanced real-time robotic simulator [19] sold by 4D-Virtualiz. This tool has been developed by two PhD students to accelerate the

development of their robotic applications. The simulation offers the ability to have ground truth, repeatability, and easy environment management with the same robot-sensor set-up. The simulated robot is a Dr. Robot Jaguar, virtually equipped with an LMS100 SICK lidar and a 640 × 480 camera (pinhole model). The simulator lidar has the same parameters as the real LMS100 (min angle, max angle, resolution, frequency, etc.). The noise added to the measures is Gaussian distributed, with parameters given by the factory data-sheet. We chose different values of the seed for the random part of the noise with the simulations. Thus, the noise varies with every simulation, resulting in slightly different localization outputs. The virtual lidar also reproduces the "rolling shutter" effect when the robot is moving. Indeed, a real lidar does not grasp all of the ranges at the same moment, a mirror rotates and measure distances sequentially with time-of-flight technology. This has also been implemented in the simulator to output the most realistic lidar data possible. Two environments have been used: the first one is an artificial "maze," a simple environment only composed of walls, and the second one is the virtual model of a real hospital located in the USA (Fig. 3). The algorithms were also tested in real world, inside a building located in Clermont-Ferrand, France. The robot used for the test was a real Jaguar robot, with a Hokuyo UTM-30-LX lidar.

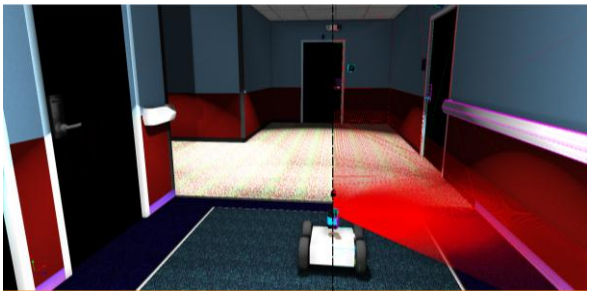

Fig. 3. Virtual realistic environment/lidar beam.

B. *Lidar Segmentation, Cascade Filter, and Scan Matching*

The cascade process is visible: as the level goes higher, there are fewer and longer segments (Fig. 4). Segments 8 and 6 have been deleted from one level to another. At the end, we have a smoothed view of the environment. Using only the scan matching between frames, we were able to perform basic localization. The robot was sent in the "maze" to do a defined trajectory, starting at a [0;0] position, and came back. The data given by the lidar were then processed by both our algorithm and the Canonical Scan Matcher (CSM) [20] provided by the ROS community [21] with the same parameters. Odometry and IMU improvement were deactivated, and we only used lidar data and constant speed assumption. The goal is not to compare complete localization methods, but rather to perform scan matching between two frames. Therefore, there is no feature recording over time, occupancy grid, nor loop closing. We were able to run autonomously the algorithm 50 times and output the trajectory, execution time and errors in scan matching (Table I). edx (edy) is the mean error in displacement toward x (y). edθ is the mean angular change error. The computer used is an XPS-9550 Dell laptop with an i5-6300HQ 2.3 GHz processor (mono-thread execution). In the end, our localization algorithm is the closest to the initial point. The total trajectory length is 110 m. Our average final error is 0.8 m, and CSM has a 1.4 m error (Fig. 5). Only 12 trajectories are shown.

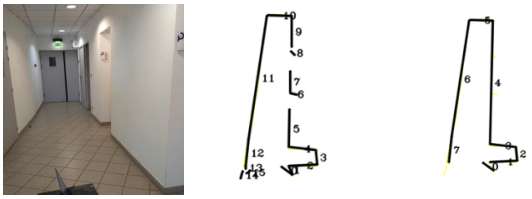

Fig. 4. Two levels of observation: left, low level; right, high level.

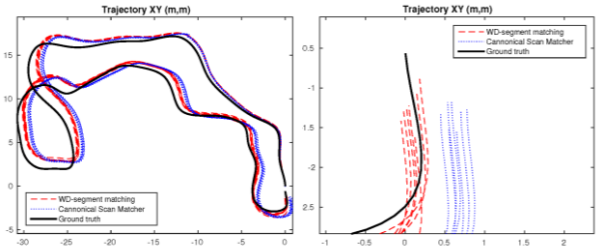

Fig. 5. Benchmark trajectories and zoom on the final error

TABLE I: BENCHMARK RESULTS

Comparison	CSM	W-D matcher
t (ms)	10.8655	0.2473
σ t (ms)	4.3691	0.0396
edx (m)	0.0898	0.0073
σ edx (m)	0.1785	0.0113
edy (m)	0.0026	0.0113
σ edy (m)	0.0739	0.0087
edθ (rad)	0.0088	0.0100
σ edθ (rad)	0.1260	0.0122

C. *Fusion with Camera*

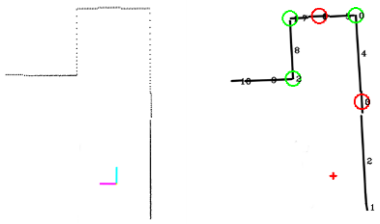

Fig. 6. Corner and door detection.

We observe an effective detection of doors and corners inside the lidar frame (Fig. 6). The robot is the red cross, doors are the red circles, and corners are the green circles. Then, we can visualize the projection of these features in the camera image (Fig. 7). We can define a box around the detected feature (corner and sides of the doors). They can become SLAM features or area of interest. In this use/case, the image had already been processed by region segmentation. The lidar works together with the camera

and gives information on the location of the area of focus. This combined work gives the boxes to process in the vision algorithm, saving time and CPU/GPU resources.

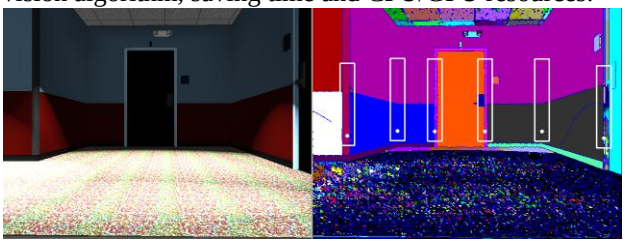

Fig. 7. Projection of the features in the camera image and segmentation

V. Conclusions

In this paper, we presented our approach to use an algorithm from another robotic field and adapt it. One goal was to exploit the structure of the environment to be able to quickly and efficiently perform scan matching for robot navigation. The execution time of our scan matcher is significantly lower than that of the other scan matcher, including the segmentation process. The computed localization with our scan matching was also more accurate. However, this requires a structured environment, which is not a requirement for every scan matcher. The other goal was to extract semantic information of this structured environment. The door and corner detector works fine with this segmentation method. The method was able to pinpoint the features in the camera image, but it did not detect doors with small shift and doors with too small angles with the lidar. These features will be included in any SLAM algorithm as new features to track and localize with. Therefore, this feature detection can help in several aspects of robot navigation: robot safety and localization. They bring localization information for the SLAM and semantic information for human detection.

VI. Future Work

Our work will focus on pedestrian anticipation, detection, and tracking. Using the area of focus given by the lidar, it should improve the computation complexity and assure a real-time detection simultaneously with the localization algorithm

Acknowledgment

This research was performed within the framework of a CIFRE grant (ANRT 2016/0316) for the doctoral work of G. Burtin at 4D-Virtualiz and LISV

References

[1] O. El Hamzaoui, “Simultaneous localization and mapping for a mobile robot with a laser scanner: CoreSLAM,” theses, Ecole Nationale Supérieure des Mines de Paris, September 2012.

[2] C. F. Chang, C. C. Tsai, J. C. Hsu, S. C. Lin, and C. C. Lin, “Laser self-localization for a mobile robot using retro-reflector landmarks,” 2003.

[3] A de Cabrol, T Garcia, P. Bonnin, and M. Chetto. “A concept of dynamically reconfigurable real time vision system for autonomous mobile robots,” *International Journal of Automation and Computing*, vol. 5, no. 2, pp. 174-184, April 5, 2008.

[4] A. de Cabrol, P. Bonnin, T. Costis, V. Hugel, and P. Blazevic, “A new video rate region colour segmentation and classification for sony legged robot application,” *Lecture Notes in Computer Science*, 2005, pp. 436-443.

[5] V. Nguyen, A. Martinelli, N. Tomatis, and R. Siegwart, “A comparison of line extraction algorithms using 2d laser rangefinder for indoor mobile robotics,” in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 1929–1934.

[6] K. Bayer, “Wall following for autonomous navigation,” SUNFEST, University of Pennsylvania, 2012.

[7] T Pavlidis, *Algorithms for Graphics ANS Image Processing*, Springer Verlag, 1982.

[8] S. I. Roumeliotis and G. A. Bekey, “Segments: A layered, dual-Kalman filter algorithm for indoor feature extraction,” in *Proc. IEEE/RSJ International Conference on* Intelligent Robots and Systems, 2000, vol. 1, pp. 454–461.

[9] R. A. Nachar, “Vers un efficace détecteur de trait: Les coins de contour et ses applications,” PhD thesis, Versailles-St Quentin en Yvelines, 2014.

[10] K. Wall and P. E. Danielsson. “A fast-sequential method for polygonal approximation of digitized curves,” *Computer Vision, Graphics, and Image Processing*, vol. 28, no. 2, pp. 220–227, 1984.

[11] F. Vincent, “Modelisation de l’environnement et localisation pour un véhicule,” Master’s thesis, L’Institut National Polytechnique de Grenoble, 1997.

[12] I. Ohya, A. Kosaka, and A. Kak, “Vision-based navigation by a mobile robot with obstacle avoidance using single-camera vision and ultrasonic sensing,” *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 969–978, 1998.

[13] Y. Hieida, T. Suenaga, K. Takemura, J. Takamatsu, and T. Ogasawara, “Real-time scan-matching using l0-norm minimization under dynamic crowded environments,” in *Proc. Fourth Workshop Planning, Perception & Navigation for Intelligent Vehicles*2, 2012, pp. 257–26.

[14] E. B Olson, “Real-time correlative scan matching,” in *Proc. IEEE International Conference on Robotics and Automation*, 2009, pp. 4387–4393.

[15] P. Vath and B. Ummenhofer, “2d multi-resolution correlative scan matching using a polygon-based similarity measurement,” Ais. Informatik. Uni., 2013.

[16] C. Berger, “Toward rich geometric map for slam: Online detection of planes in 2d lidar,” *Journal of Automation Mobile Robotics and Intelligent Systems*, vol. 7, 2013.

[17] A. Censi, “On achievable accuracy for range-finder localization,” in *Proc. IEEE International Conference on Robotics and Automation*, Roma, Italy, pp. 4170–4175, 2007.

[18] M. Alshawa, “lcl: Iterative closest line a novel point cloud registration algorithm based on linear features,” *Ekscentar*, vol. 10, pp. 53–59, 2007.

[19] G. Burtin, F. Malartre, and R. Chapuis, "Reducing the implementation uncertainty using an advanced robotic simulator," in *Machine, Control and Guidance*, 2016.
[20] A. Censi, "An ICP variant using a point-to-line metric," in *Proc. IEEE International Conference on Robotics and Automation*, Pasadena, CA, May 2008.
[21] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, *et al.*, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop on Open Source Software*, 2009.