LiDAR-based localization system for kidnapped robots

Thibaud Lasguignes, Guillaume Gobin, Olivier Stasse

LiDAR-based localization system for kidnapped robots

Thibaud Lasguignes
LAAS, CNRS
University of Toulouse,
Toulouse, France
thibaud.lasguignes@laas.fr

Guillaume Gobin
LAAS, CNRS
University of Toulouse,
Toulouse, France

Olivier Stasse
LAAS, CNRS
Artificial and Natural Intelligence Toulouse Institute,
Toulouse, France
olivier.stasse@laas.fr

***Abstract*—Place recognition is the ability to recognize previously seen places in the world. When the environment is known, the robot may localise itself when it starts and whenever it needs to verify where it is. We propose a solution to implement place recognition capability using a geometric representation of the environment build with a LiDAR. Nowadays, large Field Of View LiDARs allows to get a dense and precise representation of the environment. Assuming that a 3D map of the environment is available, we propose to build a compressed codebook based on FPFH descriptors. The codebook is then used to find the robot localization with one LiDAR image. The system has been tested on real and simulated measurements and shows promising results. The pose estimation can be thoroughly enhanced with point cloud registration methods.**

***Index Terms*—LiDAR, place recognition, kidnapped robot, humanoid robot, perception**

I. Introduction

The context of this work is the capability for a robot to localize itself in a large industrial environment such as an aeronautic factory. In this context, few textures are present and many unusual large objects are present in the environment. Still it is possible to apply a LiDAR-based localization system [1], implemented on a Talos robot [2], for allowing a precise estimation of the robot localization in a known environment. A limitation of this system is the need for an estimate of the robot initial pose. When the visual-inertial odometry or the base estimator is started, each makes an initial assumption about their world origin.

Thus, the pose ${}^{M}p_0$ of the robot in the environment when the localization system is started needs to be estimated in order to initialize the SLAM system by providing the estimation in the known world. In the experiment presented in [1], an approximation of the Mocap estimate, that was used as ground truth, was used as the initial pose.

This problem of knowing the initial pose of the robot in the environment is called the *kidnapped robot problem*. This problem is equivalent to the place recognition problem that has attracted interest in the last years. The place recognition tries to recognize a known part of the environment to help the initialization of the system or correct the drift that can be present in state estimation and odometry mechanisms, performing *loop closure*.

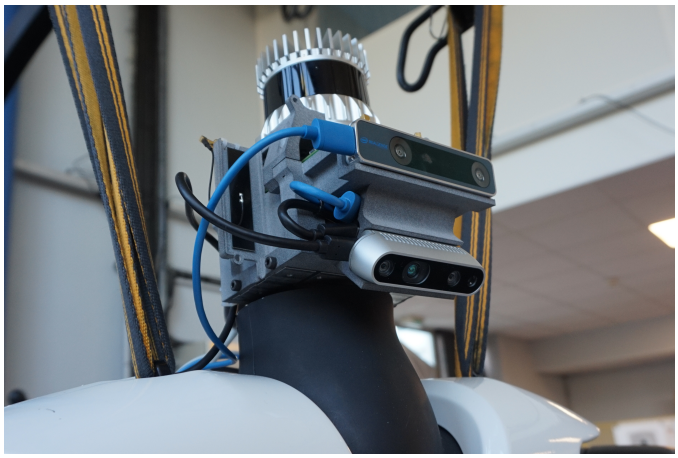


Fig. 1: New experimental head of the TALOS robot, with the D435i on the bottom, the T265i in the middle and the OS1-64 LiDAR on the top.

A. TALOS and its sensors

TALOS is a humanoid robot of 1.75m weighting 100kg with 32 degrees of freedom. It has been designed to perform complex locomotion and bi-handed manipulation involving high payload.

Initially, it was equipped with a single RGB-D camera on the head. To enhance its sensing capabilities, the head was replaced to integrate new sensors. Fig 1 shows the new head installed with a RGB-D camera Intel®RealSense™D435i [3] on the bottom, a tracking camera Intel®RealSense™T265 [4] in the middle and an Ouster OS1-64 [5] LiDAR on top.

In this work, only the LiDAR on the top of the head is used. This LiDAR has an horizontal field of view of 360° with a vertical one of 33.2° sampled by a total of 64 vertical beams. It can produce up to 1 310 720 points per second, with an accuracy of 3cm up to 60m and 10cm up to 120m.

B. Contributions

The contribution of this work are:

- Propose a solution the the place recognition problem using FPFH descriptors and VLAD encoding, with a RANSAC and ICP refinement

- Evaluate the system on simulated and real measurement using a LiDAR with a horizontal Field of View of 360°

II. Related work

As place recognition is an important problem it is widely studied topic. Its formulation dependent on the environment scale, the used sensors, the application, etc.

There are two common choices of visual sensors: camera and LiDAR. Each sensor has its pros and cons as summarized in [6]. For example, cameras have a limited Field of View (FoV) whereas nowadays LiDAR can have a 360° FoV on one axis. However, in terms of cost and energy consumption, cameras are often cheaper and less energy-consuming than LiDAR. The target environment needs to be taken into account as cameras are dependent of textures whereas LiDAR mostly relies on reflective materials. Other sensors overcome the dependance to textures of monocular cameras by being able to produce depth information, either with stereo-vision or infrared projector and cameras. However, these depth information are less precise than LiDAR's. With the evolution of technology and particularly MicroElectroMechanical Systems (MEMS), cameras integrate LiDAR systems such as the Intel-RealSense L515 LiDAR Camera [7], enhancing the range and precision of the depth information.

Depending on the sensors used, different features are used. In case of a camera, the system may be using SIFT [8] or ORB [9] whereas Scan Context [10] or FPFH [11] and others 3D descriptors may be used in case of using a LiDAR.

Other method such as in [12] try to combine both worlds instead of using only one of these visual sensors. In this work, features are extracted from the LiDAR using MinkLoc3D [13], presented on Section II-B, and from the monocular with a part of ResNet18 [14]. These two features are lately aggregated in a single one which is matched to a database.

The LiDAR-based methods may differ in the type of information processed.

A. Segment-based methods

[15] presented the concept of segments or how to use the segmentation of a point cloud to reduce its computational cost.

This concept is used by [16] and [17] to perform place recognition and SLAM. In both works, the point cloud is segmented and features are extracted for each segment. Then, using these features, the segments are matched to known ones from a map in a learning approach. These matches are then given to a geometric verification system based on RANSAC to evaluate the consistency of each match.

On this same principle, [18] applies a similar method but the feature is learnt and the final pose is estimated with PRObabilistic SAmple Consensus (PROSAC) [19] instead of the usual RANSAC method. The main advantage of their method is the design of the network that is able to run on a single CPU whereas numerous network-based method needs a GPU to be time-efficient. [20] presented an online LiDAR-based SLAM system combining the Autotuned-ICP (AICP) [21] for point cloud matching, and the Efficient Segment Matching (ESM) [18] to detect loop-closure.

B. Geometry-based methods

The main characteristic of LiDAR is the ability to provide a measure of the environment as a 3D point cloud, giving strong geometric information. Whereas methods divide the point cloud into segments, others take the complete point cloud and use its geometric information to compute local and global features.

[10] presented a new descriptor called Scan Context, a 2D-shaped descriptor used to describe 3D LiDAR scans. This descriptor divide the space into azimuthal and radial bins, similarly to [22], with each bin being represented in the descriptor by the maximum height of the points in the space division. A way to compute the similarity between the descriptors in order to be robust to the sensor's orientation is also presented in the paper. This descriptor is then used to detect loop-closure.

[23] presented an algorithm for the registration between a local point cloud and a large-scale point cloud. The system is based on the selection of local subsets, called super-points, which are described with unsupervised auto-encoders. The super-points from the local measure are matched to the ones of the large-scale point cloud acting as base. These matches are then used to make a coarse registration, later refined by an Iterative Closest Point (ICP) method.

[24] proposed PointNet, a deep neural network taking raw 3D point clouds and learning local features. The initial work focus on classification and segmentation tasks but it has opened the door of deep neural network to point cloud based systems.

[25] presented a place recognition solution based on PointNet and NetVLAD [26]. The first one is used to extract local descriptors, shortened of its maxpool aggregation layer used for its original purpose. NetVLAD is then fed with these descriptors in order to extract a global descriptor.

[13] presented an alternative approach called MinkLoc3D. This approach uses a convolutional neural network to compute local descriptors. A generalized-mean (GeM) pooling is then used to produce global descriptors.

The global descriptors obtained are used in a base-query manner. The measurement, called query, is described and compared to the descriptors available in a database.

C. Intensity-based methods

The geometry-based method may lack the appearance information available in camera-based systems. Most LiDAR provides, in addition to geometric information, intensity information that depends mostly on the texture and material of the object. Fig. 2 shows an intensity image that can be acquired with the Ouster OS1-64 set at 1024 samples per turn. Thus, works were proposed combining this intensity information with the geometric one.

[27] proposed a new descriptor, called ISHOT, combining the geometric information, represented by a SHOT descriptor [28], [29], to the intensity. This descriptor is then used in a place recognition algorithm for local descriptor evaluation and mapping.

Fig. 2: Intensity image reconstructed, the data is from an experiment of [1], it has been divided in two parts for visibility purposes.

[30] proposed to use the vision-based ORB descriptor of [9] applied to the LiDAR intensity. The ORB features are then converted into a bag-of-words vector using DBoW [31] and compared to a database.

D. Attention-based methods

Point Contextual Attention Network (PCAN) [32] is a neural network adding an attention map in the process. The system extract local features using PointNet. Then these features are fed to PCAN that output a per-point attention map, used to tune NetVLAD which is used to aggregate the local features into a global descriptor.

[33] proposed a self-attention and orientation encoding network (SOE-NET). They integrate an orientation-encoding process into PointNet to extract local features. A self-attention unit influencing the aggregation performed by NetVLAD is applied on the set of features to extract a global descriptor of the input point cloud.

III. VLAD-based solution to the kidnapped robot problem

In this section, we discuss the solution proposed. We also discuss the results obtained and some observations made during this work. More details on what is presented can be found in [34].

A. Pipeline

The proposed solution is based on the pipeline of PointNetVLAD [25]. During a first step, it creates a local descriptor and during a second step it creates a global signature. In our case, it was decided to use the FPFH at a point p as the local signature $\mathscr{L}(p)$. Two signatures were tested for the global signature $\mathscr{G}(C_s)$ of a cloud C_s. One based on the computation of a normalised sum of the local descriptors, the second one using the Vector of Locally Aggregated Descriptors (VLAD) encoding over the FPFH.

B. Local Descriptor: the Fast Point Feature Histogram

Fast Point Feature Histograms (FPFH) [11] are features proposed as an improvement of the Point Feature Histogram (PFH) [35], [36] which reduces the computational complexity.

PFH has a theoretical computational complexity for a given point cloud with n points of $O(n \cdot k^2)$ [11], with k the number of neighbors for each point p in the point cloud. FPFH reduces this computational complexity to $O(n \cdot k)$ by reducing the pairs computed.

Taking each pair (p_i, p_j) of the query point and a point in the neighborhood (p_i being the point in the pair with the shortest angle between its normal and the vector linking the points), a Darboux $u \times v \times n$ frame is defined as:

$$u = n_i, v = (p_j - p_i) \times u, w = u \times v \tag{1}$$

Then, three features are computed with:

$$\begin{aligned} f_1 &= v \cdot n_j \\ f_2 &= (u \cdot (p_j - p_i)) / \left\| p_j - p_i \right\| \\ f_3 &= \arctan(w \cdot n_j, u \cdot n_j) \end{aligned} \tag{2}$$

Each pair considered is then stored in an histogram, with the bin index defined as:

$$idx = \sum_{i=1}^{4} \left[\frac{f_i \cdot d}{f_{i_{max}} - f_{i_{min}}} \right] \cdot d^{i-1} \tag{3}$$

with []-operator being the integer part operation, d the number of subdivision of the feature's maximum theoretical value range $(f_{i_{max}} - f_{i_{min}})$. After being increased by 1, each bin's value is normalised with the total number of point pairs $(k \cdot (k+1)/2)$.

The histogram of these features are computed only on pairs composed of a given point p and its direct neighbors. This simplified histogram is called the Simplified Point Feature Histogram (SPFH).

To compute the final Fast Point Feature Histogram, the SPFH of a seek point is weighted by its neighboring SPFH values:

$$FPFH(p) = SPFH(p) + \frac{1}{k} \sum_{i=1}^{k} \frac{1}{\omega_i} \cdot SPFH(p_i) \tag{4}$$

where ω_i represents the distance between p and p_i in a given metric space.

Then the final histogram is normalized for each features for comparison purposes. In our work, the FPFH is computed with 33 bins, 11 per feature, using the implementation available in Open3D [37].

C. Global Descriptor

1) FPFH average signature: The first method defined was to group the FPFH computed on each points of the scene in a global one. We propose to define the signature in a bi-channel way composed of:

1) the normalised sum of the descriptors
2) the standard deviation of this normalised sum

Given C_s a point cloud and a normalisation function $Normalize()$, the formulations of the normalised sum and its standard deviation are respectively presented in Eq. 5 and Eq. 6.

$$\mathscr{S}(C_s) = Normalize\left(\sum_{p \in C_s} \mathscr{L}(p) \right) \tag{5}$$

$$\sigma_{\mathscr{S}}(C_s) = \sqrt{Normalize\left(\sum_{p \in C_s} \mathscr{L}(p)^2\right)} \tag{6}$$

The global signature is then:

$$\mathscr{G}(C_s) = \begin{bmatrix} \mathscr{S}(C_s) \\ \sigma_{\mathscr{S}}(C_s) \end{bmatrix} \tag{7}$$

2) VLAD encoded signature: The VLAD encoding is based on a dictionary of words. To obtain this dictionary, it has been decided to use a k-means algorithm.

Given k centroids, the k-means algorithm is performed over the sets of local descriptors of the different views from the data-base used to build the codebook. Once the dictionary obtained, a view is classified by assigning each local descriptor d to a word c_i in our codebook using a *Nearest Neighbor* search. Then, for each word of the dictionary, the distance to its affiliated descriptors in the view is aggregated over each dimension of the local descriptor. The distances for each word are concatenated and used as the VLAD descriptor. This complete encoding allows to reduce a point cloud of n points, with $n \times 3$ informations, to a descriptor of size $k \times D$, with D the dimension of the local descriptor and k the number of words in the dictionnary.

The word association is performed thanks to a *Nearest Neighbor* algorithm applied on the codebook with the distance defined as an Euclidean distance of FPFH.

The final complete pipeline is presented in Fig. 3. The circled part correspond to the process of building the memory of the system that can be made offline. The bottom part of the figure correspond to the process made online for each new request.

3) Defining the codebook: As stated in the previous section, the VLAD mechanism needs a codebook. This codebook, obtained with the k-means algorithm, presents a main parameter: the number of "words".

This number of words can be either arbitrarily defined or optimised. We tested the two possibilities, with a first codebook defined with 33 words after having tested various values of codebook. A second evaluation was performed with a codebook optimised with the "Elbow Method" [38]. This method allows to define the optimal number of words to represent a set.

For a set of data $\mathscr{D}$ to be clustered with k clusters and $\mathscr{K}_k(\mathscr{D}_i)$ the function that provides the corresponding cluster of the i-th data, the method computes the "Within-Cluster Sum of Squared errors" (WCSS) defined in eq. 8, often called *inertia* of the k-means fitting.

$$\mathscr{E}(\mathscr{D}, k) = \sum_i \|\mathscr{D}_i - \mathscr{K}_k(\mathscr{D}_i)\|^2 \tag{8}$$

Repeating this computation for multiple values of k, we obtain a convexe function of the error. Indeed, the closest the number of words is to the number of data in the set, the smallest the error is. The optimal number of words k_{opt} to represent the set is then defined as the inflection point of this function $\mathscr{E}(\mathscr{D}, k)$.

The main drawback of such a method is the computation time. To obtain the optimal number of words, the computation needs to be performed over an interval $[1; k_{max}]$ with a high enough number k_{max} to find the inflection point. To reduce this computation time, the process is performed iteratively. First the inflection point is roughly estimated by computing the WCSS at regular values over the interval $[1; k_{max}]$. Then this computation is refined by computing WCSS for more precise values around the previous estimate. This method allows to avoid computing each value in the interval while still keeping a global view of the shape of $\mathscr{E}(\mathscr{D}, k)$.

As an example, the first evaluation will be made on an interval of $[1; 501]$ with values stepped by 25 to obtain the first estimate k_{rough}. Then the same data is completed with WCSS computed around k_{rough} stepped by 1 to refine the estimation of k_{opt}.

Fig. 4 shows the Elbow method computed for the "Real" dataset. Firstly, $\mathscr{E}(\mathscr{D}, k)$ is computed with $k \in [1 : 25 : 501]$ and the method estimate the inflection point at $k_{opt,1} = 26$. Secondly, the computation is refined by computing the WCSS around $k_{opt,1}$ with $k \in [1 : 25 : 101]$. In the second step, the inflection point is evaluated at $k_{opt,2} = 16$. Lastly, the step is repeated with $k \in [1 : 1 : 31]$ and the method estimate the final inflection point at $k_{opt,3} = 31$.

D. Refining the pose estimation

The process presented above provide the estimation of the pose of the query cloud in the environment as one of the poses of the dataset. However, as the dataset is sampled over the environment, the nearest pose can be spaced from the real pose of the robot. Moreover, as neither the FPFH nor the VLAD are influenced by the rotation of the point cloud and because the LiDAR has a 360° of horizontal field-of-view, the orientation is not accurate. Therefore, we refined the pose estimated by registering the point clouds from the query and the dataset candidates.

This registration is performed with two steps: a coarse estimation using RANdom SAmple Consensus [39] and a refinement using a Point-to-Plane ICP [40] method. The RANSAC method is applied using the FPFH computed for the VLAD encoding to estimate correspondences. Then the coarse estimation is used to initiate the locally convergent ICP.

This registration is successively performed on the n top candidates from the VLAD-encoding recognition, n being defined arbitrarily for the tests. Later, the finer alignment, using the alignment scores defined in the original work for each method, is used to give the estimated pose.

E. Datasets

There are 2 datasets that were produced. The first one was produced using a simulated version of the Bauzil Room. A second dataset has been built in a real setup. Fig. 5 shows the real Bauzil Room in 2021 and a simulated version viewed from opposed points of view.

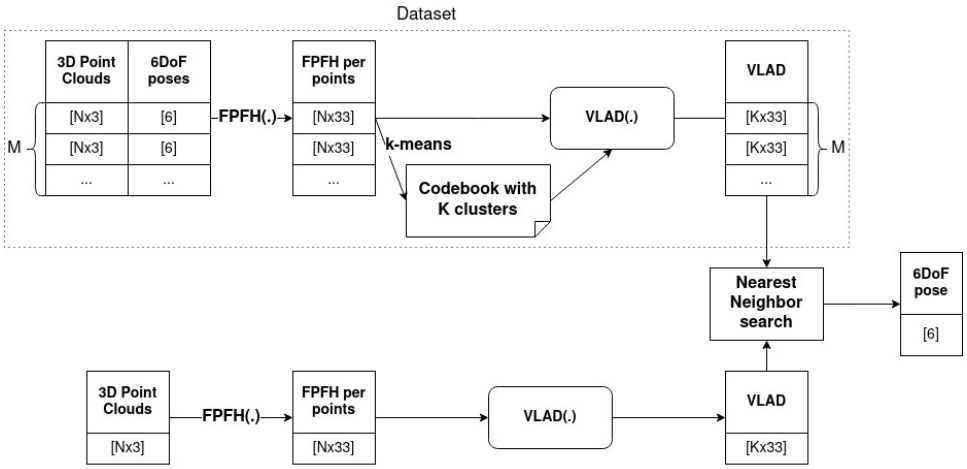


Fig. 3: Architecture of the VLAD-encoding recognition system.

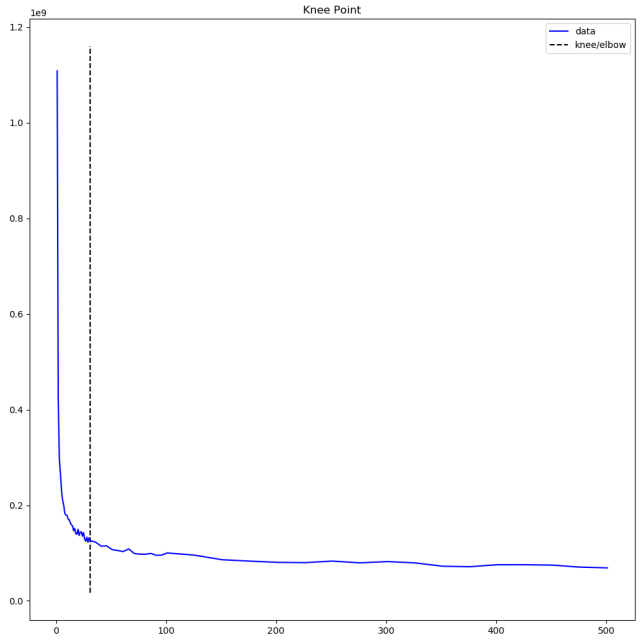
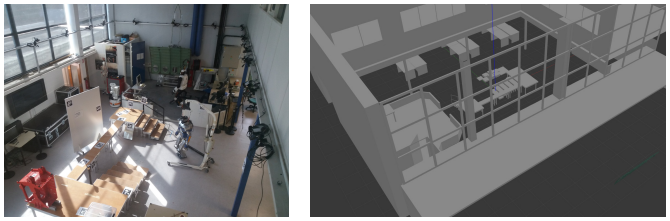


Fig. 4: Computation of the Elbow Method for the "Real" dataset.



(a) Real (b) Simulation

Fig. 5: The Bauzil room and its simulated version in Gazebo.

1) Simulated sets: For the simulated experiments, the middle platforms with the stairs were removed from the reality and the simulation. The simulation uses a modeled LiDAR on the architecture and noise-model of the Ouster OS1-64. The data-base has been generated with the robot placed along a (x,y) grid. The query-base has been built with the robot randomly placed in the room, covering the same (x,y) space as the data-base. Both bases are generated with the sensor at a constant height of 1.65m, the approximate heigh of the sensor on the robot when in its half-sitting pose. Fig. 6 shows examples of the distribution of the poses from the simulated dataset. In the figure, the bases sizes have been reduced for visualization purposes. Each smaller axis represent a pose present in the database. The data-base shown is generated over a 8×11m grid graduated every meter. The query-base is generated with 50 random poses from the same 8×11m space.

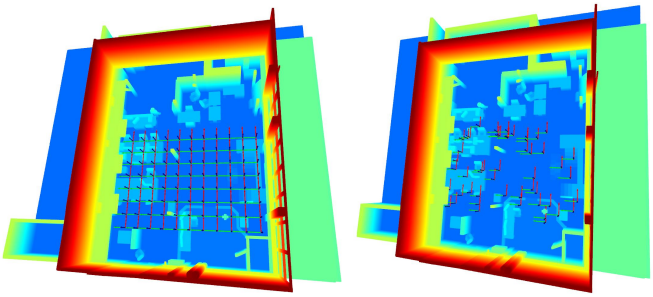
In the generation of the simulated data-bases and query-bases, the orientation is left aside. Indeed, the LiDAR has a 360° horizontal field of view. Moreover, when we start the robot, it is assumed that the LiDAR (x,y) plane is parallel to the world's (x,y) plane.

In the case of a noiseless sensor, the data is really smooth and the FPFH descriptors are highly discriminating as shown in Fig. 7 by the clustering of the FPFH in 4 groups. In this example, the FPFH were computed with a 50cm radius and a limit of 4000 used neighbors.

2) Reality sets: The robot has been put in different places of the Bauzil room with its poses estimated using the available Motion Capture system. Thus, the zone measured has been limited by the zone covered with the Mocap cameras. Again, the measures for the data-base were taken with an approximate 1m grid on (x,y) axes. Whereas the query-base is taken with random (x,y) positions. Moreover, the clouds were captured on positions where the robot was standing still for 10s to avoid noise from the robots movements.

IV. Experimental Results

This solution has been evaluated on three VLAD codebook created using the proposed method. Two were build using the real and the simulated datasets, and are respectively



(a) Data-base (b) Query-base

Fig. 6: Visualization of the poses generated distribution. The number of poses is reduced for visualization purposes.

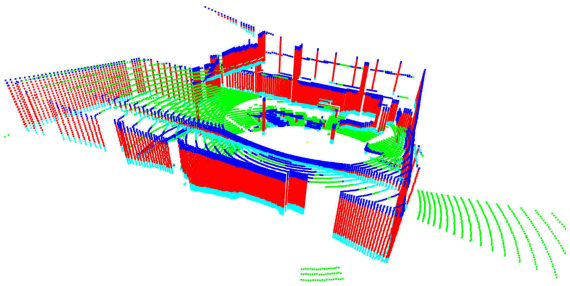


Fig. 7: Visualization of a simulated cloud clustered in 4 groups.

called "Real" and "Simulated". The third one was created from merging the real and simulated dataset and is called "Combined". Each dataset consists of a data-base and a query-base. The experiments are summarized in Table I.

Experiment	Data-base	Clusters	Query-base	Queries
Simulation	*Simulated*	100	*Simulated*	500
Reality	*Real*	31	*Real*	77
Combined	*Simulated+Real*	136	*Simulated +Real*	577

TABLE I: Data-base and Query-base for the tests performed, with their respective number of clusters in the codebook of the data-base and queries in the query-base.

Both pose estimations steps are evaluated: the VLAD-encoding recognition, the RANSAC coarse registration and the ICP refinement.

A. VLAD-encoding registration

To define the results, it is proposed to compare 3 positions, visible in Fig. 8: The *Ground Truth (GT)* position of the robot, the *Nearest* position existing in the data-base, considered as the *must give* from the system, and the *Recognized* position given by the system. A test is then considered as correct if the system has given the *Nearest* position as the *Recognized* one.

Table II summarize the percentage of correct matches in the top candidates, according to the previous statement.

We can observe that the system succeed poorly on the top candidate for the "Real" but the good recognition stays mainly

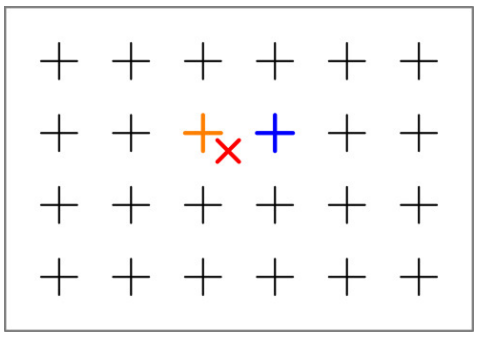


Fig. 8: Visualization of the 3 proposed positions: the *Ground Truth* in red, the *Nearest* in orange and a potential *Recognized* one in blue, other positions in the data-base in black.

Top candidates considered	"Simulated"	"Real"	"Combined"
1	81.60%	42.86%	76.26%
2	95.00%	68.83%	87.18%
3	97.60%	77.92%	91.16%
4	99.80%	84.42%	92.89%
5	100.00%	89.61%	93.24%

TABLE II: Percentage of correct recognitions with the VLAD-encoding depending on the number of top candidates. (FPFH radius: 0.5 meter, FPFH maximum neighbors: 4000)

in the 5 top candidates. For the "Simulated" and "Combined" tests, we can observe that in 3 tests out of 4, the nearest pose in the data-base is given as the top 1 candidate. Going down to the 5-th candidate rise significantly the percentage of success.

The results for both cases are promising for the efficiency of the RANSAC and ICP refinement as there are high probability to register two close data.

B. RANSAC coarse estimation

For the RANSAC and ICP estimations, the error evaluated is the final translation error and the rotation error, as a global rotation and as Roll-Pitch-Yaw errors. A case is considered successful if the error in translation is below 10cm and the error in rotation is below 5°. These thresholds have been fixed relatively to the problematic: initializing the localization system presented in [1]. We experimentaly determined that the localization mechanism needs an initialization with 50cm of translation tolerance and 30° of rotation in the $\vec{z}$-axis.

Table III present the percentage of queries for which the system respects either one of our criteria or both, using only the RANSAC refinement. The results shows that RANSAC has a high level of success for the orientation estimation. However, the translation criteria is respected in less than half the cases.

Table IV shows statistics of the translation error for each cases. It is observable that for both tests, the mean translation error is not far from the criteria. Table V presents the statistics of the rotation error. We can see that the mean error is of 2°, below the threshold defined.

Test	"Simulated"	"Real"	"Combined"
Success on translation error	36.20%	36.36%	41.94%
Success on rotation error	99.00%	96.10%	98.61%
Global success	36.20%	36.36%	41.94%

TABLE III: Percentage of successful estimations with the RANSAC refinement only.
(FPFH radius: 0.5 meter, FPFH maximum neighbors: 4000)

Test	Mean	Median	Min	Max	RMS	STD
"Simulated"	12.69	11.37	1.53	58.69	14.44	6.89
"Real"	14.45	12.28	2.14	43.44	17.00	8.95
"Combined"	12.49	10.88	0.88	133.55	15.11	8.49

TABLE IV: Translation error of the pose estimation using RANSAC, in cm.
(FPFH radius: 0.5 meter, FPFH maximum neighbors: 4000)

Test	Mean	Median	Min	Max	RMS	STD
"Simulated"	0.0214	0.0170	0.0005	0.2715	0.0295	0.0203
"Real"	0.0348	0.0306	0.0053	0.0843	0.0401	0.0199
"Combined"	0.0211	0.0154	0.0001	0.2720	0.0304	0.0219

TABLE V: Rotation error of the pose estimation using RANSAC, in rad.
(FPFH radius: 0.5 meter, FPFH maximum neighbors: 4000)

C. ICP refinement

Table VI present the percentage of queries for which the system respects either one of our criteria or both, using and ICP refinement over a RANSAC coarse estimation. The results show an improvement over the RANSAC algorithm. Whereas the rotation criteria is less respected, the percentage of success on the translation criteria is higher.

Test	"Simulated"	"Real"	"Combined"
Success on translation criteria	95.20%	70.13%	96.53%
Success on rotation criteria	98.20%	97.40%	98.79%
Global success	95.20%	70.13%	96.53%

TABLE VI: Percentage of successful estimations with the RANSAC+ICP refinement.
(FPFH radius: 0.5 meter, FPFH maximum neighbors: 4000)

Tables IV and V summarize respectively the translation and rotation statistics on the RANSAC and ICP refinement. These results shows that while the percentage of success is higher, the failure errors are also higher, with a maximal translation error at least multiplied by 6.

Test	Mean	Median	Min	Max	RMS	STD
"Simulated"	14.40	8.96	3.75	617.14	46.80	44.53
"Real"	14.17	8.71	6.51	303.66	37.76	35.00
"Combined"	12.43	8.92	0.46	696.54	43.42	41.60

TABLE VII: Translation error of the pose estimation using RANSAC and ICP, in cm.
(FPFH radius: 0.5 meter, FPFH maximum neighbors: 4000)

1) Time consumption: The time consumption of the system has been observed. The computation have been performed on a computer, using an Intel®Core™i5-8400H CPU at 2.50GHz with 32GB of RAM.

Test	Mean	Median	Min	Max	RMS	STD
"Simulated"	0.0219	0.0006	0.0000	4.3512	0.2758	0.2750
"Real"	0.0233	0.0186	0.0030	0.2891	0.0422	0.0352
"Combined"	0.0214	0.0006	0.0000	4.3721	0.2575	0.2566

TABLE VIII: Rotation error of the pose estimation using RANSAC and ICP, in rad.
(FPFH radius: 0.5 meter, FPFH maximum neighbors: 4000)

The FPFH computation and VLAD encoding took a mean time of 10s with 95% of the time allocated to the FPFH computation.

The coarse estimation using the RANSAC algorithm took a mean time of 40s, using the FPFH descriptors previously computed. However, depending of the randomisation, the range of time consummed in the RANSAC estimation goes from 20s to 90s.

Lastly, the ICP refinment, took a mean time of 10s taking as initialisation the RANSAC estimation.

Thus, a complete query took a mean time of 1min10s with longer computations rising up to 2min for a single query.

This time consumption is high but acceptable in the case of a single computation when the robot is started or of sparse computation over the time to relocalise the robot in long runs.

V. Conclusion

We presented a pipeline to recognize the place from where a LiDAR measurement is taken in a known environment. It is supposed that a quantity of data from the environment taken with the sensor is available to create a database to refer to. The system uses the FPFH descriptor to describe locally the data and a VLAD-encoding to describe the complete measurement. This FPFH+VLAD-encoding is used to compare a new query to the known database, allowing to propose candidates from the database that are close to the request. These candidates are then used to perform a point cloud registration using a Point-to-Plane ICP initialized by a RANSAC estimation.

The systems has been tests with simulated data and measurements taken with a LiDAR in the real environment. The VLAD-encoding recognition shows a percentage of success close to 90% in the 5 best candidates. Using these candidates, the RANSAC and ICP refinement allows to localize the measurement with a mean error of less than 15cm in translation and less than 2° in rotation.

The results shown are of the order of magnitude of those obtained with deep neural network-based methods such as PointNETVLAD [25] with 80% of success. However, deep networks are runned on GPU with really fast estimations, in range of $5-10$ms, whereas our solution has only been tested on CPU with a slower computation time. In future works, this system will be deployed on the robot and verified on new environments and databases from the state of the art such as [41] or [42] to be compared to other solutions.

ACKNOWLEDGEMENTS

This work was supported by the cooperation agreement ROB4FAM. The use of the experimental platform TALOS was supported by ROBOTEX 2.0 (Grants ROBOTEX ANR-10-EQPX-44-01 and TIRREX-ANR-21- ESRE-0015).

REFERENCES

[1] T. Lasguignes, I. Maroger, M. Fallon, M. Ramezani, L. Marchionni, O. Stasse, N. Mansard, and B. Watier, "Icp localization and walking experiments on a talos humanoid robot," in *Int. Conf. on Advanced Robotics (ICAR)*, pp. 800–805, 2021.

[2] O. Stasse, T. Flayols, R. Budhiraja, K. Giraud-Esclasse, J. Carpentier, J. Mirabel, A. Del Prete, P. Souères, N. Mansard, F. Lamiraux, J.-P. Laumond, L. Marchionni, H. Tome, and F. Ferro, "TALOS: A new humanoid research platform targeted for industrial applications," in *IEEE-RAS Int. Conf. on Humanoid Robotics (ICHR)*, 2017.

[3] Intel, "Intel® realsense™ depth camera d435i." https://www.intelrealsense.com/depth-camera-d435i/.

[4] Intel, "Intel® realsense™ tracking camera t265." https://www.intelrealsense.com/tracking-camera-t265/.

[5] Ouster, "High-resolution os1 lidar sensor: robotics, trucking, mapping." https://ouster.com/products/scanning-lidar/os1-sensor/.

[6] T. Barros, R. Pereira, L. Garrote, C. Premebida, and U. J. Nunes, "Place recognition survey: An update on deep learning approaches," *arXiv preprint arXiv:2106.10458*, 2021.

[7] Intel, "Intel® realsense™ lidar camera l515." https://www.intelrealsense.com/lidar-camera-l515/.

[8] D. Lowe, "Object recognition from local scale-invariant features," in *IEEE Int. Conf. on Computer Vision (ICCV)*, vol. 2, pp. 1150–1157 vol.2, 1999.

[9] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *IEEE Int. Conf. on Computer Vision (ICCV)*, pp. 2564–2571, 2011.

[10] G. Kim and A. Kim, "Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map," in *IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 4802–4809, Oct. 2018.

[11] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *Int. Conf. on Robotics and Automation (ICRA)*, pp. 3212–3217, 2009.

[12] J. Komorowski, M. Wysoczańska, and T. Trzcinski, "Minkloc++: Lidar and monocular image fusion for place recognition," in *Int. Joint Conf. on Neural Networks (IJCNN)*, 2021.

[13] J. Komorowski, "Minkloc3d: Point cloud based large-scale place recognition," in *IEEE Winter Conf. on Applications of Computer Vision (WACV)*, pp. 1789–1798, 2021.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[15] B. Douillard, A. Quadros, P. Morton, J. P. Underwood, M. De Deuge, S. Hugosson, M. Hallström, and T. Bailey, "Scan segments matching for pairwise 3d alignment," in *Int. Conf. on Robotics and Automation (ICRA)*, pp. 3033–3040, May 2012. ISSN: 1050-4729.

[16] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, "Segmatch: Segment based loop-closure for 3d point clouds," *Int. Conf. on Robotics and Automation (ICRA)*, pp. 5266–5272, May 2017. arXiv: 1609.07720.

[17] R. Dubé, A. Cramariuc, D. Dugas, J. Nieto, R. Siegwart, and C. Cadena, "Segmap: 3d segment mapping using data-driven descriptors," *Robotics: Science and Systems (RSS)*, June 2018. arXiv: 1804.09557.

[18] G. Tinchev, A. Penate-Sanchez, and M. Fallon, "Learning to see the wood for the trees: Deep laser localization in urban and natural environments on a cpu," *IEEE Robotics and Automation Letters (RAL)*, vol. 4, pp. 1327–1334, Apr. 2019.

[19] O. Chum and J. Matas, "Matching with prosac - progressive sample consensus," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 220–226 vol. 1, 2005.

[20] M. Ramezani, G. Tinchev, E. Iuganov, and M. Fallon, "Online lidar-slam for legged robots with robust registration and deep-learned loop closure," in *Int. Conf. on Robotics and Automation (ICRA)*, pp. 4158–4164, 2020.

[21] S. Nobili, R. Scona, M. Caravagna, and M. Fallon, "Overlap-based icp tuning for robust localization of a humanoid robot," in *Int. Conf. on Robotics and Automation (ICRA)*, pp. 4721–4728, 2017.

[22] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 24, no. 4, pp. 509–522, 2002.

[23] G. Elbaz, T. Avraham, and A. Fischer, "3d point cloud registration for localization using a deep neural network auto-encoder," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 2472–2481, IEEE, July 2017.

[24] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 652–660, 2017.

[25] M. A. Uy and G. H. Lee, "Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, May 2018. arXiv: 1804.03492.

[26] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 5297–5307, 2016.

[27] J. Guo, P. V. K. Borges, C. Park, and A. Gawel, "Local descriptor for robust place recognition using lidar intensity," *arXiv:1811.12646 [cs]*, Nov. 2018. arXiv: 1811.12646.

[28] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *European Conference on Computer Vision (ECCV)*, pp. 356–369, Springer, 2010.

[29] F. Tombari, S. Salti, and L. Di Stefano, "A combined texture-shape descriptor for enhanced 3d feature matching," in *2011 18th IEEE international conference on image processing*, pp. 809–812, IEEE, 2011.

[30] T. Shan, B. Englot, F. Duarte, C. Ratti, and D. Rus, "Robust place recognition using an imaging lidar," *arXiv:2103.02111 [cs]*, Mar. 2021. arXiv: 2103.02111.

[31] D. Galvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics (T-RO)*, vol. 28, pp. 1188–1197, Oct. 2012.

[32] W. Zhang and C. Xiao, "Pcan: 3d attention map learning using contextual information for point cloud based retrieval," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 12436–12445, 2019.

[33] Y. Xia, Y. Xu, S. Li, R. Wang, J. Du, D. Cremers, and U. Stilla, "Soe-net: A self-attention and orientation encoding network for point cloud based place recognition," *arXiv:2011.12430 [cs]*, Nov. 2020. arXiv: 2011.12430.

[34] G. Gobin, "Localisation d'un robot humanoïde TALOS par information géométrique et visuelle combinée à l'apprentissage," Master's thesis, Toulouse 3 Paul Sabatier, Aug. 2021.

[35] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 3384–3391, 2008.

[36] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz, "Learning informative point classes for the acquisition of object model maps," in *Int. Conf. on Control, Automation, Robotics and Vision (ICARCV)*, pp. 643–650, 2008.

[37] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

[38] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," *Jour. of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001.

[39] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, pp. 381–395, June 1981.

[40] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and vision computing*, vol. 10, no. 3, pp. 145–155, 1992.

[41] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset," *Int. Jour. of Robotics Research (IJRR)*, vol. 36, no. 1, pp. 3–15, 2017.

[42] Y. Liu, W. Gao, and Z. Hu, "A large-scale dataset for indoor visual localization with high-precision ground truth," *Int. Jour. of Robotics Research (IJRR)*, vol. 41, no. 2, pp. 129–135, 2022.