http://dx.doi.org/10.1109/IROS.2016.7759504

Long-term place recognition using multi-level words of spatial densities

Renan Maffei[1] Vitor A. M. Jorge[1] Vitor F. Rey[1] Mariana Kolberg[1] Edson Prestes[1]

***Abstract*— Proper place recognition on an environment that can change over time is fundamental for long-term SLAM. In such scenarios the observations obtained in the same region can drastically differ due to changes caused by semi-static objects, such as doors, furniture, etc. In this work, we extend a strategy that represents environment regions using words, based on spatial density information extracted from laser readings. This time, in order to deal with changes in the environment, our method not only builds words representing the real observations made by the robot, but also alternative multi-level words to account for possible changes in a place's observations generated by non-static objects. Place recognition is made by searching matches of sequences of N consecutive words (both real or alternatives). Experiments performed in real and simulated scenarios are shown, and demonstrate the advantages associated to the use of multi-level words.**

I. Introduction

Long-term mobile robot operation is a field that has attracted the interest of many robotics researchers [1], [2], [3]. With the increasing integration of robots in human activities, it is essential that they have the ability to adapt to changes in the environment. For example, a robot equipped with a laser range finder moving inside a building must differentiate static objects, such as walls and columns, from highly dynamic objects, such as people in motion. It must also differentiate both types of objects from semi-static objects, such as doors, tables, and other furniture, which can move occasionally.

Although most of the Simultaneous Localization and Mapping (SLAM) techniques work well in static environments and many techniques are able to deal with highly dynamic environments (e.g. using methods for filtering moving objects in the robot's field-of-view [4]), identifying semi-static objects is a very difficult problem. This comes from the fact that a robot cannot distinguish between static objects or dynamic objects that are standing still during a single visit to a place. Thus, revisiting an environment that suffered changes in the disposition of its objects will lead to conflicting sensor measurements, which can rather hinder the localization and mapping process. Nonetheless, such revisiting behavior is required in order to keep an updated map.

Some SLAM techniques deal with dynamic environments by dividing the problem into building two distinct maps: one for static objects and other for dynamic objects [5], [6]. However, in long-term operations, objects can move in highly different time scales. Some authors propose a representation with multiple maps, where each one is updated in a specific time scale [1], [2], others propose a single dynamic occupancy grid, which stores, for each grid cell, a probabilistic estimate of changes in the environment [7], [8]. A more recent approach is the dynamic pose graph (DPG) [3], which is updated whenever changes in the environment occur, by inserting or removing nodes or connections between nodes. An important aspect of DPG is that it maintains a history of changes in the environment, instead of just keeping an updated current map.

An assumption made by most long-term SLAM approaches is that the starting and ending point of the robot trajectory are known. In situations where this is not the case (e.g. when we simply start the robot navigation from an unknown place), the data association problem becomes much more difficult. A successful place recognition is fundamental for SLAM. In fact, the occurrence of a few false positives can fully degrade the solution [9], [10].

During the last years, various approaches for place recognition in dynamic scenarios have been presented in the literature, mostly based on computer vision and using a wide range of different strategies to find matches, such as template matching, feature extraction, Bag of Words, etc [11], [12], [13]. On the other hand, place recognition based only on laser measurements suffers from the high level of ambiguity associated with the sensors information. For instance, a robot that is moving inside a structured environment can obtain the same information in multiple occasions at different places, and thus the chance of producing false matches is large. In cases like this, the only way to disambiguate similar regions is by considering sequential information. Many works show that exploring sequences of observations during the place recognition process leads to better results than matching single observations [14], [13].

In our previous work [15], we introduce a place recognition method based on sequences of low dimensional observations acquired by a robot using a laser range finder. Such information is obtained by computing kernel density estimates (KDE) of the free space [16]. The method quantizes the information into density classes, and generates simple words to represent regions of the environment. Place recognition is made by matching sequences of words. As shown in [15], the method obtains good results in structured environments and can perform fast searches for long sequences of observations, however, it was focused on static environments.

Our new approach takes the idea of using words to represent regions and expands it for the problem of long-term operation. When an environment is not static, we cannot assure that all information been observed it will always be constant. What if a word that was just created belongs to a

[1]Institute of Informatics, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil rqmaffei, vamjorge, vfrey, mariana.kolberg, prestes@inf.ufrgs.br

We thank the financial funds provided by CNPq and CAPES.

dynamic object? Then we must consider the possibility of not finding this specific word when revisiting this specific region. Therefore, the core of our proposal is that every time a word is built, we also try to build alternative words that would exist in the absence of the observed one. That is, when building a new word, we also consider the case where the density change was brought by a non-static object and create a new word sequence assuming the previous density remained present in the new word's location.

This paper is presented as follows. Section II presents the algorithm for building words and finding matches as introduced in our previous work [15]. Section III details the expansion of the method for long-term place recognition. Section IV presents a density-based strategy that we use for fine tuning of matches. Section V shows the results of experiments in long-term operation. Lastly, in Section VI we draw our conclusions and present future work.

II. Representing Trajectories using N-grams of Spatial Densities

A robot equipped with a laser range-finder that moves through structured environments obtains a series of consistent signatures associated to spatial regions. As proposed in our previous work [15], we can translate the sequence of such signatures into simple words, reducing the complexity of the data association problem. The matching of n-grams[1] can be performed in an efficient way, while the matching of the corresponding metric region using a traditional point matching method, such as ICP, can be orders of magnitude slower [15]. In this section, we present a brief description of the original method, which was fully described in [15].

The first step of the algorithm is obtaining the signature of the current spatial region by computing the local kernel density estimate Ψ [16] of free space in the robot position. In order to compute this density, we maintain a local grid map, as show in Fig. 1. The KDE Ψ of free space surrounding point $\mathbf{p}_0 = (x_0, y_0)$ is computed through Eq. 1

$$\Psi(\mathbf{p}_0) = \sum_{\mathbf{p}} s(\mathbf{p}) K(||\mathbf{p} - \mathbf{p}_0||), \tag{1}$$

where $\mathbf{p}$ is a point in the map, $K(.)$ is a kernel profile used to restrict the surrounding grid cells, and $s(\mathbf{p})$ is a function that selects only points detected as free space. The idea is that densities associated to narrow spaces, such as corridors (e.g. point $\mathbf{p_1}$ in Fig. 1), will be consistently smaller than densities associated to wide spaces, such as bifurcations (e.g. point $\mathbf{p_0}$ in Fig. 1).

Next, in order to build words, density signatures (Ψ) are quantized into density classes, which can be made by uniformly partitioning the density value's space. A word is associated to each contiguous sequence of observations from same density class. Words must be larger than a certain threshold t_o, otherwise the presence of any kind of noise, such as small moving obstacles, would hurt the matching process.

[1]An n-gram is an ordered sequence of size n. A gram can be a word (as in our case), a syllable, a letter, etc. n-grams are used for language identification, string matching, among other applications.

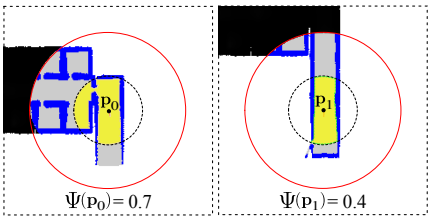


Fig. 1: Example of spatial density values. The red circle delimits the frontier of the local grid map, the black circle delimits the region under the kernel influence and the yellow region corresponds to the visited space covered by the kernel.

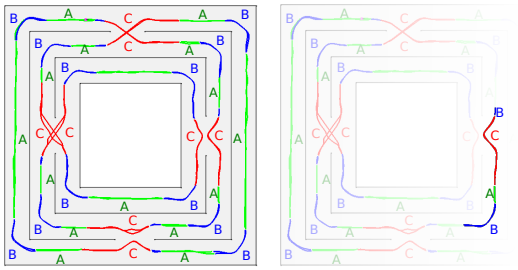
Algorithm 1 shows the process of word construction. The algorithm receives a list O of consecutive observations from same density class, where each observation $o = \{\Psi, (x, y)\} \in O$ has a density signature (Ψ) and a position (x, y) given by odometry. The information associated to O is translated into a word composed of three syllables. The first syllable, D, is the density class of all observations, obtained by a quantization function q. The second syllable, S, is the number of observations that compose the word. The last syllable, A, is the difference of the angle between the median and the final position of the sequence of observations, and the angle between the initial and the median position. Finally, we also associate the list of observations to the word to store the location of such word in the robot trajectory.

Algorithm 1: Build Word

Input: $O = [o_i, \cdots, o_{m=(i+k)/2}, \cdots, o_k]$
Output: w
1 $w.D = q(\Psi_{o_i})$
2 $w.S = size(O)$
3 $w.A = atan\left(\frac{y_{o_k} - y_{o_m}}{x_{o_k} - x_{o_m}}\right) - atan\left(\frac{y_{o_m} - y_{o_i}}{x_{o_m} - x_{o_i}}\right)$
4 $w.obs = O$
5 **return** w

The place recognition strategy is performed by comparing sequences of n words – or n-grams – over the entire trajectory. A match between two grams is considered successful only if there are matches between all i-th words of both grams. In the same way, the matching between two words considers all three syllables. While the matching considering the density syllable is exact (i.e. either the words are in the same density class or they aren't), the matching considering the size and angle syllables have pre-defined tolerances (empirically determined as 25% and $30°$, respectively).

Fig. 2 shows an example of words construction. In (a), we can see the densities quantization, which separates the environment in three different types of regions – corridors, corners and bifurcations. Four consecutive words are presented in (b), along with their descriptions in (c). For instance, the last highlighted word $(B, 18, -90)$ corresponds to a segment of density class B, composed of 18 observations, and an



(a) (b)

Words:		2-Grams:	3-Grams:
B,4,0	C,23,-5	(B,4,0; C,23,-5)	(B,4,0; C,23,-5; A,7,0)
A,7,0	B,18,-90	(C,23,-5; A,7,0)	(C,23,-5; A,7,0; B,18,-90)
		(A,7,0; B,18,-90)	

(c)

Fig. 2: Example of words construction. (a) Density quantization into 3 classes. (b) Highlighting 4 consecutive words. (c) Description of the highlighted words, and the corresponding 2-grams and 3-grams.

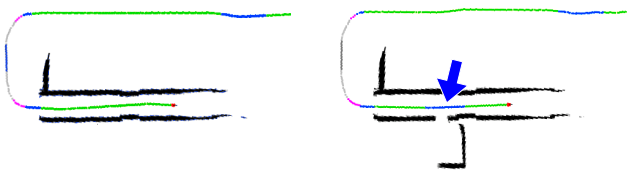
angle variation of $-90°$ (i.e. a turn to the right). Fig. 2(c) also shows the 2-grams and 3-grams which can be associated to those four words.

III. Long-term place recognition using multi-level words of spatial densities

When the robot is in long-term operation, changes in the environment affect the density estimates, damaging the performance of the word matching. Fig. 3 exemplifies a case where the robot moves alongside a door that is closed in the first moment, (a), and open in the second moment, (b). When this situation occurs, instead of building a single large word to represent the region, the method will create two median-sized words of the same density class separated by a small word of a different density class. However, the size of the original large word is similar to the combined size of the three smaller words.

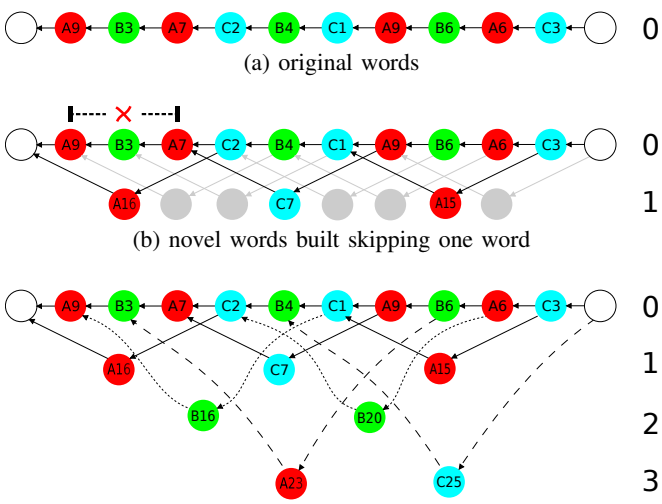
That being said, the core of our long-term place recognition technique is generating "corrected" words for all regions. Basically, we start with the case where one word is wrong, such as the open door in Fig. 3(b), and expand it to cases where multiple consecutive words are wrong. Therefore, the algorithm works in different levels, where the number of levels is equal to the number, L, of words which will be replaced to generated a single word. In detail, the algorithm detects sequences of L+2 words, i.e. (L+2)-gram, which start and end with the same density class but have L words in between them with significantly different density classes. Those sequences are then fused into a larger word having the same density class. For instance, when the robot is visiting the region in Fig. 3(b), the method will generate three words for the *green-blue-green* segment of the trajectory, but it will also generate a larger single *green* word and associate it to the same segment.

Algorithm 2 presents the strategy for building multi-level words, while an example of multi-level words is presented in



(a) Closed door (b) Open door

Fig. 3: Changes in the environment, such as a open/closed door, affect the word construction. The higher density region (in blue, pointed by the blue arrow) is only obtained when the door is open.



(a) original words

(b) novel words built skipping one word

(c) all novel words built skipping one, two and three words

Fig. 4: Example of multi-level word construction. The densities of the first and last word of the sequence must match in order to replace the density of the middle words and build a new word. Arrows point to the predecessors of each word.

Fig. 4. The input for the algorithm is the list of consecutive observations O, along with the current lists of words in each level, $W^0, W^1, \cdots W^L$, where $W^l = [w_0^l, w_1^l, \cdots w_n^l]$ with w_j^l being the j-th word of the l-th level. The first step is building the current 0-level word, adding it to the corresponding list, and adding the previous 0-level word to its list of predecessors words (lines 1-4). Note that a word can have multiple predecessors (one word from each level).

Next, for each l upper levels, we check if the last 0-level word (w_{n-1}^0) and the 0-level word that is $l+1$ positions before it (w_{n-l-2}^0) have the same density class (line 6). For instance, to add a new word to level 1, the words w_{n-1}^0 and w_{n-3}^0 must have the same density class. If this is the case, the new high-level word is computed using the concatenation of all observations associated to the last $l+2$ words (lines 7-9). For example, in Fig. 4(b), the initial three words – of sizes 9, 3 and 7 – generate a word of size 16 with the same density class of the first and third ones. Fig. 4(c) shows all possible words that can be generated in this example using three additional levels of words.

Lastly (lines 10-12), the list of predecessors for the new high-level word is copied from the word at 0-level having

the same start position, given that they share the same predecessors. Additionally, the new high-level word is added to the predecessors list of the last 0-level word.

Algorithm 2: Building Multi-level Words

Input: $O, W^0, W^1, ..., W^L$
Output: $W^0, W^1, ..., W^L$

$w = BuildWord(O)$
$W^0.add(w)$
$n = size(W^0)$
$w^0_n.pred = [w^0_{n-1}]$
for l *in* $1...L$ **do**
  if $w^0_{(n-l-2)}.D == w^0_{n-1}.D$ **then**
    $O' = concat(w^0_{n-l-2}.obs, \cdots, w^0_{n-1}.obs)$
    $w = BuildWord(O')$
    $W^l.add(w)$
    $m = size(W^l)$
    $w^l_m.pred = w^0_{n-l-2}.pred$
    $w^0_n.pred.add(w^l_m)$
return $W^0, W^1, ..., W^L$

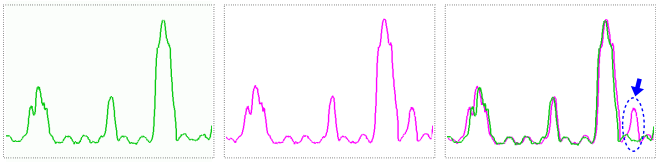
After updating the words, we perform the place recognition by searching n-grams combining those multi-level words. The search is incremental regarding the levels of words, i.e., at first it considers only the lowest level, then it considers the two lowest levels, and so on. This is made to ensure that when we check the l-th level, we will at least obtain the matches of the $l-1$ levels.

In each iteration, we initially check for matches of the last words of all levels. We maintain separated lists of word occurrences to speed up the process, so that this initial search can be made without sweeping the whole trajectory every time. Following this one-word match, we just extend the search visiting the predecessor's lists of the matched words. Given the strategy's multi-level aspect, we may obtain multiple matches with different n-grams at each step. For instance, one specific n-gram can be matched simultaneously with an n-gram composed of words from the low-level, with another one composed of words from an upper-level, and, most probably, with an n-gram mixing different levels.

Like in our previous work [15], we set a minimum number of words that must be reached to accept a match. Small n-grams sizes lead to large number of matches, but potentially low precision, while large ones give rise to low recall. Thus, determining good thresholds is an important aspect to take into consideration. Moreover, we must use different thresholds for different levels. Note that matches in the order of 10 or 15 words are less likely to occur in the lowest level than in upper levels, because the solution space increases in the upper levels.

IV. Fast Adjustment of Matches by Evaluating Raw Spatial Densities

The focus in [15] was on finding matches which were topologically corrected. While this remains the core of our current proposal, this time we also deepen the investigation of translating our n-grams matches into more precise metric matches. When considering a topological view, simply matching regions regardless of size would imply that they have the same importance, which is just adding one true correspondence to the total. However, the impact of matching different sizes of regions drastically influences the quality of a final metric map.



(a) closed door (b) open door (c) alignment

Fig. 5: Variation of raw density values (Ψ) obtained in the regions shown in Fig. 3. In (c), all values match well, with exception of the small bump associated to the open door.

In order to capture this type of information, we translate all the matches of words to equivalent sets of points, which are generated in constant intervals based on the odometry. When a match occurs between two words, the regions represented by those words are converted into subsets of the existing set of points that must be connected. Finding a proper connection between points is important, and thus we propose a fine tuning method using raw densities values.

When we quantize the raw densities signatures to build words, we are simplifying the information to facilitate the matching of large trajectory segments. However, we can still compare the raw density values of the observations associated to the words to obtain better similarity measures. Therefore, we search the best alignment Δ between two sets of observations that minimizes the sum of the squared error between density values.

$$\Delta = \underset{-\Delta_{Max}<d<\Delta_{Max}}{\arg\min} \sum_{i=0}^{n} \left(o_{(s1+i+d)}.\Psi - o_{(s2+i)}.\Psi\right)^2, \quad (2)$$

where $s1$ and $s2$ are the starting observations of both words, n is the size of the smallest word between the two, and Δ_{Max} is the maximum displacement that we are still searching.

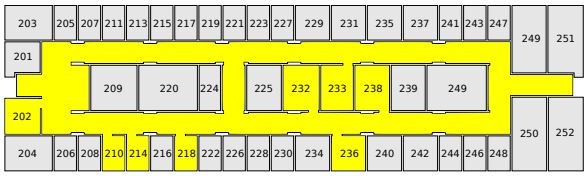
Fig. 5 shows an example of raw density values from the matched regions presented in Fig. 3. As we can observe, the two curves of density values are similar, with the only significant difference occurring in the exact position associated to the open door (as pointed by the blue arrow). Still, considering that changes in the environment are generally small and somewhat distant from each other, such alignment method can obtain good results.

V. Experiments

The evaluation of our method was made using a Pioneer 3-DX mobile robot equipped with a SICK LMS 200 laser range finder. The robot performed multiple runs starting from different positions in the environment shown in Fig. 6, which is an indoor environment in our university. All tests were made considering the same configuration for generating

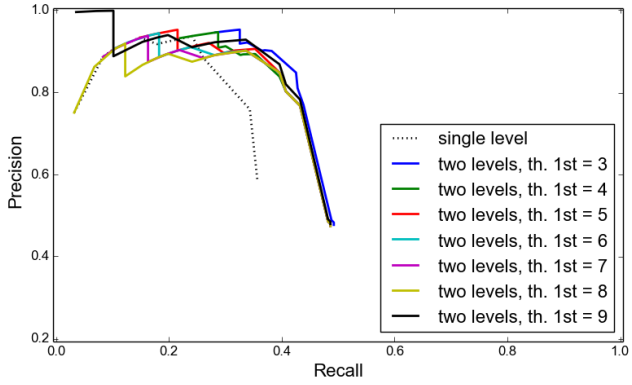
Fig. 6: Environment used in the experiments.

(a) low variation

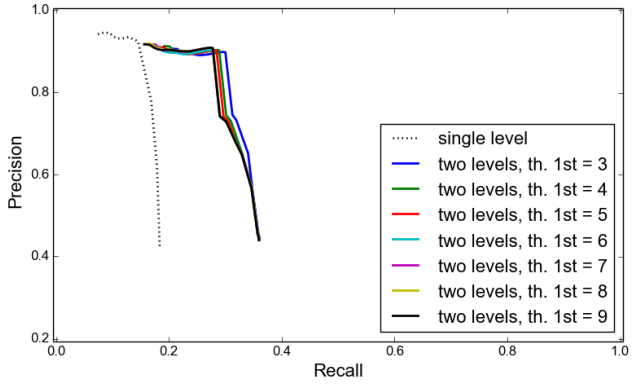


(b) high variation

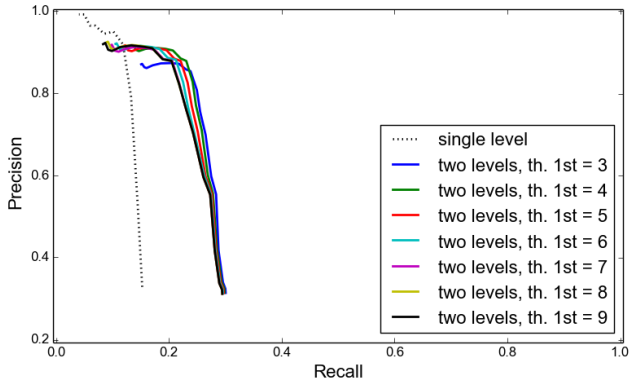
Fig. 7: Examples of environment configurations. (a) Configuration used in scenario B – we change at most one door per corridor, usually in the middle of them. (b) Configuration used in scenario C – we change adjacent doors, doors near corners, etc.



(a) Scenario A



(b) Scenario B



(c) Scenario C

Fig. 8: *Precision/Recall* plots obtained in each scenario for the single level approach and for two-levels approaches fixing the threshold of the first level and varying the thresholds of the second level.

words: kernel radius of $2.5m$ for estimating densities, 8 classes of densities during quantization, 25% and $30°$ as size and angle thresholds during the syllables generation.

Three scenarios were chosen to evaluate the performance of the method: one real and two simulated generated using the MobileSim Simulator. Scenario A is the real unsupervised scenario, which we do not have any control on the dynamics of the environment, such as open/closed doors, or people passing by the robot. Scenario B is a simulated scenario with low changes in the environment from one run to the others. Particularly, we only alter from the configuration with all doors closed (as shown in Fig. 6) to two configurations with a single open door for each corridor (an example is shown in Fig. 7(a)). Scenario C is a simulated scenario with frequent changes in the environment. We alter from the closed doors configuration to other five with multiple doors opening in the same corridor or opening near corners (an example is shown in Fig. 7(b)). The robot performed, in different days and hours, around 8 laps in Scenario A, and around 40 laps in Scenarios B and C.

In a first moment, we compared the performance of single level runs against two-levels runs. In the single level runs, we just varied the minimum size of grams for accepting matches (from 0 to 9). In the two-levels runs, we fixed the threshold for the first level in different configurations (from 3 to 9) and varied the threshold for the second level (from 0 to 18). Fig. 8 shows plots of precision/recall for the three scenarios. Given that our search for matches only happens in the same direction of the robot trajectory, we decided to consider in the recall calculation only matches that could be obtained in the same direction. All results show that, with similar precision, the recall of multi-level approaches (solid lines) is higher than the recall of single level approaches (dotted lines).

Analyzing the individual results, all strategies obtained the highest recalls in Scenario A. This is somewhat expected because A is the shortest scenario, meaning that the main variations are caused by people walking, which are small in

TABLE I: Comparisons of precision, recall and time (to process one lap in the environment).

	Precision - Recall			Time(s)
	Scenario A	**Scenario B**	**Scenario C**	
1-level	0.905 - 0.103	0.931 - 0.106	0.964 - 0.070	2.5
2-levels	0.910 - 0.238	0.903 - 0.274	0.912 - 0.169	11
3-levels	0.935 - 0.346	0.890 - 0.308	0.913 - 0.188	38
ICP	0.048 - 0.040	0.047 - 0.011	0.219 - 0.040	96

comparison to opening and closing doors. On the other hand, the lowest recalls were obtained in Scenario C. This was also expected because C is the most dynamic scenario, and it has situations that our current algorithm does not cover well, such as density variations occurring near corners. Making an analogy to Fig. 5, a variation occurring near corners do not produce a density bump in the middle of a homogeneous region, but a density bump over another existing density bump, which is more difficult to detect.

Finally, we evaluate the running time of the proposed method. One of the main advantages of our previous work is its high speed to compute place recognition in all trajectory [15]. As expected, the use of multiple levels of words to improve recall makes the approach slower. Table I shows the comparison among a single level run (with threshold 6), a two-level run (with thresholds 6 and 9), a three-level run (with thresholds 6, 9 and 12) and ICP (using 150 points extracted from regions created at each 20 steps). Each level that is added, reduces the speed of the method, however, even with three levels of words the method is faster than a traditional point matching technique such as ICP with few points. And comparing the results of precision and recall, the proposed method is able to obtain much higher precision than ICP, because it can match very long segments of the trajectory in an efficient way.

VI. Conclusion

In this paper we propose a strategy for long-term place recognition by matching sequences of words built from spatial density information. We can detect changes in the environment by creating multiple words associated to the same region. That means, given a density variation observed in the robot trajectory, our method builds a word to represent such variation and build other words to represent what the robot should be seeing if such variation did not exist. Experiments were made in real and simulated scenarios of an indoor environment, and have demonstrated improvements in the matching results.

The main contributions of the approach are:

(i) A novel representation of dynamic regions associating words of spatial density in multiple levels;
(ii) A fast strategy for finding matches in sequences of multi-level words;
(iii) A fine-tuning adjustment of matches using raw kernel density estimates of the free space.

We have observed that detecting changes in the environment near places with high variations of densities is still a problem for the method. At this moment, the method only deals with variations that are preceded and succeeded by densities of the same class. Thus, we are studying ways of generating alternative words for dynamic objects in situations like that.

Finally, our current method only searches for matches in the same direction that was visited by the robot, since it is based on n-grams, i.e. ordered sequences of words. As future work, we also want to generate reversed words, and find matches using such information.

References

[1] P. Biber and T. Duckett, "Dynamic maps for long-term operation of mobile service robots," in *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2005.

[2] ——, "Experimental analysis of sample-based maps for long-term slam," *The International Journal of Robotics Research*, vol. 28, no. 1, pp. 20–33, jan 2009.

[3] A. Walcott, M. Kaess, H. Johannsson, and J. J. Leonard, "Dynamic pose graph slam: Long-term mapping in low dynamic environments," in *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Piscataway, NJ, USA: IEEE Press, Oct 2012.

[4] D. Hahnel, R. Triebel, W. Burgard, and S. Thrun, "Map building with mobile robots in dynamic environments," in *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2. Piscataway, NJ, USA: IEEE Press, sept. 2003, pp. 1557–1563.

[5] D. F. Wolf and G. S. Sukhatme, "Mobile robot simultaneous localization and mapping in dynamic environments," *Auton. Robots*, vol. 19, no. 1, pp. 53–65, jul 2005.

[6] C.-C. Wang, C. E. Thorpe, S. Thrun, M. Hebert, and H. F. Durrant-Whyte, "Simultaneous localization, mapping and moving object tracking," *I. J. Robotic Res.*, vol. 26, no. 9, pp. 889–916, 2007.

[7] G. Tipaldi, D. Meyer-Delius, M. Beinhofer, and W. Burgard, "Simultaneous localization and dynamic state estimation in reconfigurable environments," in *IEEE/RSJ IROS Workshop on Metrics and Methodologies for Autonomous Robot Teams in Logistics (MMART-LoG)*, San Francisco, USA, October 2011.

[8] G. D. Tipaldi, D. Meyer-Delius, M. Beinhofer, and W. Burgard, "Lifelong localization and dynamic map estimation in changing environments," in *Proc. of Robotics: Science and Systems 2012 Workshop on Robots in Clutter: Manipulation, Perception and Navigation in Human Environments*, 2012.

[9] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)*, ser. Intelligent robotics and autonomous agents. Cambridge, MA, USA: MIT Press, aug 2005.

[10] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, winter 2010.

[11] M. Cummins and P. Newman, "Fab-map: Probabilistic localization and mapping in the space of appearance," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[12] D. Galvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, Oct 2012.

[13] M. J. Milford and G. F. Wyeth, "Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, May 2012, pp. 1643–1649.

[14] M. Bosse and R. Zlot, "Map matching and data association for large-scale two-dimensional laser scan-based slam," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 667–691, 2008.

[15] R. Maffei, V. A. M. Jorge, V. F. Rey, G. S. Franco, M. Giambastiani, J. Barbosa, M. Kolberg, and E. Prestes, "Using n-grams of spatial densities to construct maps," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, Sept 2015, pp. 3850–3855.

[16] R. Maffei, V. A. M. Jorge, V. F. Rey, M. Kolberg, and E. Prestes, "Fast monte carlo localization using spatial density information," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, May 2015, pp. 6352–6358.