LSH-RANSAC: An Incremental Scheme for Scalable Localization

LSH-RANSAC: An Incremental Scheme for Scalable Localization

Kenichi Saeki Kanji Tanaka Takeshi Ueda

Abstract— **This paper addresses the problem of feature-based robot localization in large-size environments. With recent progress in SLAM techniques, it has become crucial for a robot to estimate the self-position in real-time with respect to a large-size map that can be incrementally build by other mapper robots. Self-localization using large-size maps have been studied in litelature, but most of them assume that a complete map is given prior to the self-localization task. In this paper, we present a novel scheme for robot localization as well as map representation that can successfully work with large-size and incremental maps. This work combines our two previous works on incremental methods, iLSH and iRANSAC, for appearance-based and position-based localization.**

I. MOTIVATION

Self-localization is an important task for mobile robot environments [1]–[3]. With recent progress in SLAM techniques, it becomes possible for a robot to obtain and use large-size environment maps that are incrementally built by other mapper robots [4]. For instance, such an online information sharing network attracts much interest in recent years, for example, in the context of networked robots, sensor networks as well as robot GIS. As a result, it has become crucial for a robot to estimate the self-position in real-time with respect to such *incremental maps*. An important challenge is self-localization using *large-size maps* where increasing number of self-position hypotheses as well as perceptual aliasing becomes a serious problem [1], [5], [6].

In this paper, we study the self-localization problem from a perspective of large-size and incremental maps. Self-localization using large-size maps have been studied in literature. However, most of them consider familiar or static environments and rely on a *batch assumption*, where a complete map is given a priori and fixed during the self-localization task. In such a situation, one can build the map structure in a batch manner using typical optimization techniques. On the other hand, we consider more general environments where one can update or incrementally build the map even during the self-localization task. In such a situation, typical batch algorithms are no longer relevant. To deal with the problem, we will present a novel scheme for self-localization as well as map representation that can successfully work with large-size and incremental maps.

This work was partially supported by MECSST Grant in-Aid for Young Scientists (B) (17700200, 19700192), by Suzuki Foundation Research Grant, by Electro Mechanic Technology Advancing Foundation, and by TATEISI Science And Technology Foundation.

K. Saeki and K. Tanaka are with Graduate School of Engineering, Fukui University, Japan. T. Ueda is with Graduate School of Engineering, Kyushu University, Japan. kanji@rc.his.u-fukui.ac.jp

II. INTRODUCTION AND RELATED WORK

Approaches to self-localization can be broadly classified into appearance-based [6]–[8], position-based [1], [3], [9] or the combination of both [10]–[12]. Appearance-based approaches efficiently prune pose hypotheses by using an appearance database, which associates the appearance of an observed landmark with its location. Position-based approaches verify each hypothesis based on the geometric constraints among observed landmarks. In large-size problems, both approaches play important roles to deal with large number of hypotheses and constraints.

In appearance-based localization, high dimensional features (e.g. SIFT [13], spin image [14], shape context [15]) are used to represent appearance features. For instance, 120-dim shape features from laser scanners are considered in this paper's experiments. To deal with the high-dimensionality, database techniques (e.g. PCA with kd-tree [11], visual vocabulary with inverted file and words statistics [8], vocabulary with ANN and feature selection [6]) are usually employed. From our perspective, a major inconvenience of typical database techniques is their batch structure as well as expensive pre-processing. On the other hand, we use hash tables as a basis of map structure (”*hash maps*”). Unlike other structures such as trees, a hash table is essentially an incremental structure. In [16], we developed incremental maps based on locality sensitive hashing (iLSH) [17].

In position-based localization, a difficulty is the computational complexity which grows exponentially with the number of landmarks. A solution to the difficulty is RANSAC map-matching [1], whose objective is to find reliable coordinate-transformation between a local map built by the robot-self and the global map built by a mapper robot. In [1], large-scale map-matching has been achieved with purely position-based RANSAC by Neira, Tardós and Castellanos [1]. In [18], map-matching has been robust even when the ratio of outlier observations is high. From our perspective, a major inconvenience of existing map-matching is that they are essentially batch algorithms, assuming that map features are fixed during the matching task. On the other hand, we pose the self-localization as an *incremental map-matching* problem. In [19], we proposed incremental RANSAC (iRANSAC) scheme and found iRANSAC is effective even for incremental maps [19], [20].

In this paper, we combine the advantages of above two methods, iLSH and iRANSAC, and develop a novel scheme called LSH-RANSAC. The main contribution of the proposed scheme is that it achieves self-localization with large-size and fully incremental maps. We experiment the scheme in large-size environments with over 10 mapper robots by using radish datasets [21].

Self-localization has been a central research area in robotics. It is difficult here to cover all the literature, including pose tracking techniques using Kalman filters. We review most relevant works in the following.

In position-based localization, another possible approach is to generate and track multiple hypotheses of the robot pose over time [3], [5], [9]–[12], [16], [19], [22]. Especially, particle filter is a popular approach that has received considerable attentions in recent years [10], [11], [22]. The methods distribute a number of pose hypotheses covering the robot's environments, then track and evaluate the likelihood of each hypothesis. In [16], we also developed a particle filter algorithm combined with an appearance-based database. From our perspective, a major drawback is that they consume a significant amount of computational costs, in principle, linear to the environment size. Scalable extension is an interesting topic of on-going research [5].

In object recognition, the combination of LSH and RANSAC also has been recently studied for large-size problems [23]–[25]. There are important applications including image retrieval [23], epipolar geometry [24] and shape retrieval [25]. However, only a few use LSH as an incremental database [26]. To our knowledge, incremental extension of LSH-RANSAC as well as application to SLAM & localization is novel.

In place recognition, visual dictionaries are recently used to interpret a high-dimensional feature to a 1D visual word [6]. However, such a dictionary is conditioned on a specific feature type, parameter setting as well as training environments. Naturally, there can be a risk of overfitting and overgeneralization. Incremental building of vocabulary [27] (e.g. k-means reclustering) is not a trivial problem [28], [29]. Our approach is rather similar with approaches in [30], [31] for object recognition in that we directly discretize the feature space without any dictionary.

In SLAM, localization using an incremental map is studied in the context of loop closure. However, most of them rely on prior knowledge (e.g. probability distribution) obtained from typical pose-tracking in SLAM. Often, the robot pose is simply represented as a 1D space of visited place ID instead of the full pose space (e.g. 3D) considered in this paper.

In computer vision, RANSAC is a standard algorithm for robust estimation [32]. To speed up for large problems, a series of randomized RANSAC algorithms (rRANSAC) as well as probabilistic variants and batch optimization of rRANSAC [25] have been studied in recent years [24], [25], [33]–[35]. The objective of most rRANSAC (e.g. $T_{d,d}$ test [34]) is similar with original RANSAC in a sense that they iterate the hypothesize-and-test until some reliable hypotheses are found. On the other hand, preemptive RANSAC (pRANSAC) proposed by Nister [35] deals with real-time applications where the computation time is always limited and typically constant. This property is appealing for real-time problem of robot localization. Our iRANSAC is an incremental extension of the pRANSAC scheme.

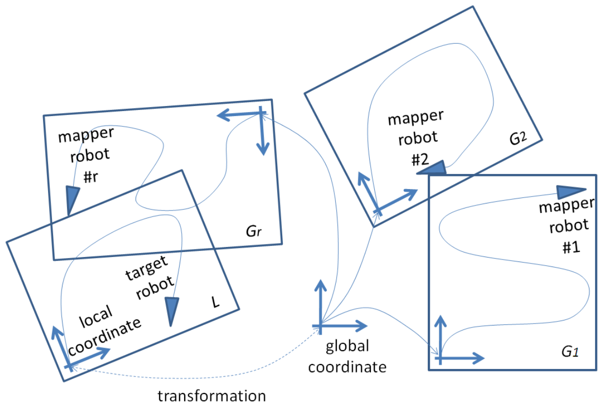


Fig. 1. Map matching problem. The objective is to estimate the coordinate-transformation betweeen the local map built by the target robot and the global map (a set of submaps) built by mapper robots.

III. Map-Matching Strategies

This section explains our basic strategies for the map-matching problem in the RANSAC formulation. Let L denote a local map. The map is built by the robot-self with its motion measurements (e.g. odometry) and perceptual measurements (e.g. laser scan) using map-building techniques such as FastSLAM [36] as well as scan matching [37]. Let G denote a global map. The global map is a set of R metrical submaps $G = \{G_1, \cdots, G_R\}$ built by R mapper robots and provided via some information sharing network [4]. The maps represent the configuration of landmarks or features in the individual robots coordinates. Fig. 1 illustrates the relationships among robots and maps. The objective of map-matching is to find a reliable coordinate-transformation ψ (translation, rotation) from the local map L to the global map G by which the two maps maximally overlap.

A. *RANSAC Map-Matching*

The general RANSAC algorithm is described as follows. $U = \{x_i\}$ denotes a set of N data-points. $f_m : S \rightarrow p$ denotes a model function that computes the model parameters p from a sample set S of data-points. $c(p,x)$ denotes the score function for a single data-point x with model parameter p, and takes 1 if x is an inlier to the model or 0 otherwise. The objective is to find a "best" model parameter p^* (with its associated cost value C^*) that should maximize the score function. The RANSAC algorithm iterates the following steps until some reliable hypotheses are found.

1) Select randomly a set $S_k \subset U$ and compute a model parameter hypothesis $p_k = f_m(S_k)$.
2) Compute score $C_k = \sum_{x \in U} c(p_k, x)$.
3) If $C_k > C^*$ then $C^* \leftarrow C_k$, $p^* \leftarrow p_k$.

It is possible to directly apply the RANSAC algorithm to a *batch* map-matching problem. In the setting, the above parameters are interpretted as follows:

- N : the number of features on the local map L,
- x : a feature on the local map L,

- p : a transformation from the local map L to the global map G,
- $c(p,x)$: the score function takes 1 if a feature point x on the transformed map L^p matches a feature point on the global map G or 0 otherwise. Here, L^p is the local map transformed by p.
- f_m : the model function generates a candidate transformation from the local map to a global map based on some local matches.

B. pRANSAC Scheme

pRANSAC maintains the history $H = \{(p_j, x_j, C_j)\}$ of every feature-hypothesis pair (p_j, x_j) scored so far and its corresponding scoring result $C_j = c(p_j, x_j)$. It also employs two user-defined functions, the order rule

$$(p,x) = f_o(H) \tag{1}$$

that decides which pair should be scored next based on the history H of scoring results, and the preference rule

$$p^* = f_p(H) \tag{2}$$

that decides which hypothesis is best based on H. Suppose we have been given a set of features

$$U = \{x_i\} \tag{3}$$

and already generated a set of hypotheses

$$V = \{p_j\}. \tag{4}$$

pRANSAC iterates the following steps until the computation time is exhausted.

1) Select a feature-hypothesis pair $(p_k, x_k) = f_o(H)$ and compute the score $c(p_k, x_k)$.
2) Update the history H based on the scoring results.

If necessary, it outputs the best hypothesis $p^* = f_p(H)$.

C. Breadth-First Rule

The performance of pRANSAC depends largely on the order rule $f_o(H)$. The order rule proposed by Nister, called preemptive breadth-first rule has some appealing properties [35]. Firstly, its computation time is bounded by a constant proportional to the size of hypothesis set. Secondly, it compares the hypotheses against each other, rather than against some absolute quality measure. Thirdly, its characteristics have been studied analytically using an inlier-outlier model. In the following, we briefly summarize the preemption rule.

To save the computation time, it employs a decreasing preemption function

$$f_b(i), \tag{5}$$

and limits the number of hypotheses to be considered (active hypotheses). It randomly permutes the sequence of feature ids $1, \cdots, N$ as well as randomly permutes the sequence of hypotheses ids $1, \cdots, M$. It initializes the score $C_j = 0$ for every hypothesis $j (1 \leq j \leq M)$. It performs the following steps for each iteration i until the computation time is exhausted.

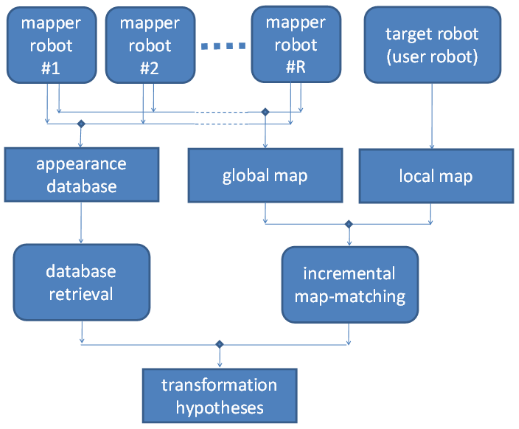


Fig. 2. Incremental map-matching system. The appearance database, the global map and the local map can be incrementally updated by robots during the map-matching task.

1) Compute the scores $C_j \leftarrow C_j + (p_j, x_i)$ for each hypothesis $j (1 \leq j \leq f_b(i))$.
2) Reorder the hypotheses ids $1, \cdots, M$ so that the range $1, \cdots, f_b(i)$ contains the best $f_b(i)$ active hypotheses according to the accumulated score C_j.

The preemption function $f_b(i)$ is in the form:

$$f_b(i) = \lfloor M 2^{-\lfloor \frac{i}{B} \rfloor} \rfloor, \tag{6}$$

where $\lfloor \cdot \rfloor$ denotes downward truncation, and B is a preset constant ("block size" [35]). Note that the size of hypothesis set reduces to the half every B^{th} iteration. This results in an approximately $O(BM)$ time cost.

IV. Incremental LSH-RANSAC Scheme

So far, we considered *batch* map-matching task where the feature and the hypothesis sets are fixed. This is not the case for the real-time task of robot localization [3], [5], [12]. To see this, we briefly describe a typical localization task in the following.

During the localization task, the localization process (i.e. the map-matching process) continues to search for answer by incorporating the latest sensor measurements. The localization process is initialized only once at the beginning of the localization task at the start viewpoint and starts with empty sets. There are two types of events by which the sets are modified:

1) When new local features arrive,
   - the local map is modified with each feature,
   - the appearance database is queried for each feature,
   - new hypotheses are generated from the retrieval result,
   - the hypothesis set is modified with the new hypotheses,
2) When new features arrive in a global map,
   - the global map is modified with each feature,
   - the appearance database is updated with each feature.

Considering the above, we need to take two points into account. Firstly, the problem size becomes intractable even by the efficient pRANSAC with $O(BM)$ complexity as the number M of hypotheses grows constantly in an unbounded fashion. To deal with this, we adapt an appearance-based technique to prune a potentially large number of new hypotheses into a tractable size (e.g. 1,000). In particular, we will use the incremental LSH (iLSH) technique to realize an incremental appearance database that can be built from scratch and incrementally updated by mapper robots. Secondly, the assumption of fixed feature and hypothesis sets is violated as the local and the global maps can be incrementally updated. To deal with this, we adapt our incremental RANSAC (iRANSAC) [19] scheme to incorporate the new features in local and global maps during the map-matching task. In particular, we will develop an incremental extension of the preemptive breadth-first rule so that the number of surviving hypotheses (or active hypotheses) are pruned into a tractable size in a preemptive manner.

In the following, we explain the methods to implement the above idea in more detail (Fig.2).

A. Appearance Database

The appearance database employs an approximate near neighbor technique called locality sensitive hashing (LSH) [38]. LSH uses a continuous representation of data-point, and searches points that are near from the query point in l_2 space. The E^2LSH algorithm addresses an (R, cR)-Near Neighbor problem, where the goal is to report a point within distance cR from a query q, if there is a database point within distance R from q [39]. E^2LSH is $O(n^{\rho}(c) \log n)$ time complexity where $\rho(c) < 1$ is an approximation factor. In some applications, LSH is shown to be significantly faster than other fast ANN algorithms [39].

Unlike other structures such as tree, structures based on hash tables have strong advantages in *incremental* settings where insertion/deletion of database elements is frequently required. In the pre-processing of E^2LSH, L hash functions with K dimensions are probabilistically generated using a p-stable distribution (e.g. the normal distribution). When a new feature arrives in global maps, the database is updated in the following procedure:

- memorizes the real-world location of the feature,
- hashes the feature using the E^2LSH function and accesses the corresponding bins,
- associates the real-world location of the feature to the accessed bins.

In our LSH system, the high-dimensional feature itself is not memorized on the database and also not used in the retrieval task. This is a simple modification but quite effective to significantly reduce the space and time costs of a database [39]. In [16], we have applied this technique to Monte Carlo localization and reported its performance. Note that the above process for updating the database (learning and dimensionality reduction) can be performed in a completely incremental manner.

We also adapt the *hash maps* [16]. The real-world locations of features are commonly memorized on grid maps. In large-size environments, naive implementation of a grid map consumes a significant amount of memory, in proportional to the number of grid cells. To deal with this, we adapt universal hashing (UH) technique that maps the real-world space (e.g. 2D or 3D space) onto 1D hash table. The size of hash table should be determined so that the probability of collision is sufficiently low taking into account the spatial density of features. Moreover, there is a technique to incrementally extend a hash table when the current hash table is found to be too small. In consequence, the space cost of such an approximated grid map is less than 3% of the original grid map in our experimental settings.

B. Map-Matching Process

The iRANSAC algorithm is summarized as follows. At the beginning of the task, it initializes the feature and the hypothesis sets to empty sets $U \leftarrow \phi$, $V \leftarrow \phi$. During the task, it iterates the following steps:

1) When new features U^{new} arrive, modify the feature set
$$U \leftarrow U^{new} \cup U, \tag{7}$$
2) When new hypotheses V^{new} arrive, modify the hypothesis set
$$V \leftarrow V^{new} \cup V, \tag{8}$$
3) Perform the steps 1,2 of pRANSAC.

The modified breadth-first rule performs the following steps for each iteration i:

1) Perform the step 1,2 of the iRANSAC algorithm,
2) Perform the step 1,2 of the original breadth-first rule.

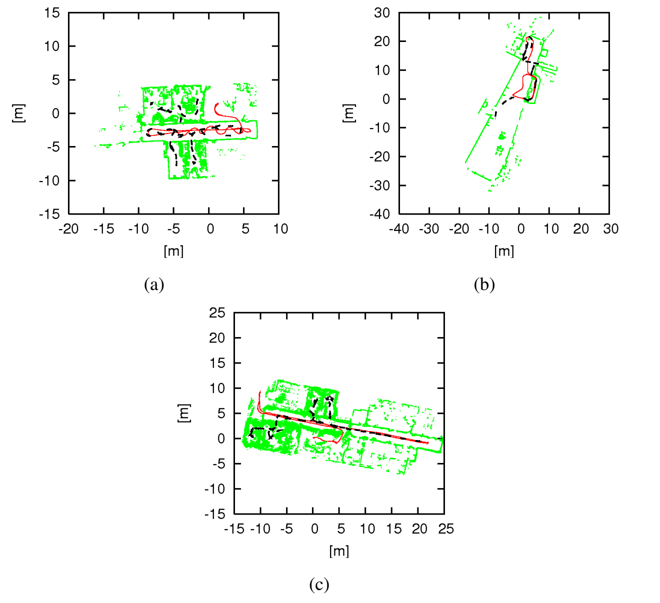
The preemption function $f_b(i)$ is in the form:

$$f_b(i) = \begin{cases} m(i-1)/2 & (i \bmod B = 0) \\ m(i-1) & (i \bmod B \neq 0) \end{cases}, \tag{9}$$

where $m(i)$ denotes the number of hypotheses at the end of i-th iteration.

The preference rule is slightly modified. In the incremental setting, the number N_j of scored can be very different among individual hypotheses. Younger hypotheses tend to be scored smaller times than older ones. For fair comparison, we use a normalized measure
$$C_j / N_j \tag{10}$$
(in place of the original measure C_j) to compare among the hypotheses. In addition, a young hypothesis whose score is yet smaller than a threshold C_o is not considered as a candidate of the best hypothesis.

The modified breadth-first rule degenerates to the original breadth-first rule in a special case where the hypothesis set is fixed. It is known that the performance of rRANSAC is maximized when a randomly permutated sequence is used [35]. However, it is no longer possible to permute the hypothesis set beforehand in the incremental setting. The process of random permutation of a set U (or V) can be implemented as a $O(|U^{new}|)$ process of inserting each element in U^{new} at a random point in U because the U has been already permuted.



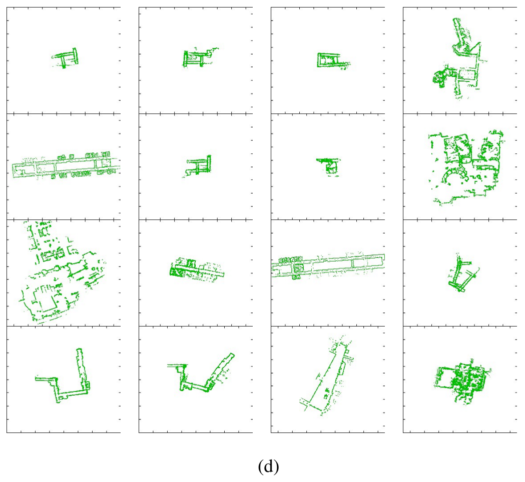


Fig. 3. Experimental environments. (a),(b),(c) The target robot's building. The solid and the dotted curves respectively are the trajectories of the target robot and the mapper robot. (d) The individual buildings including the target robot's building.

V. Experiments

We evaluate our approach through robot localization experiments using radish datasets [21] (Fig. 3). For each dataset, there are sequences of motion and perception measurements. Each motion measurement indicates the robot's movement from one viewpoint to the next and represented in a forward-sideward-rotate (FSR) format in our system. Each perception measurement is a single scan by the front laser scanner and represented by a set of 180 data-points in a robot centric coordinate. In the case of laser data-points, shape feature has been considered an appropriate type of appearance feature [23]. In particular, we use a shape feature called generalized shape context (GSC) [15] that is found to be stable and useful for robot localization applications in our previous works [16], [20]. We employ a simple scan matching algorithm as a SLAM method. The number of

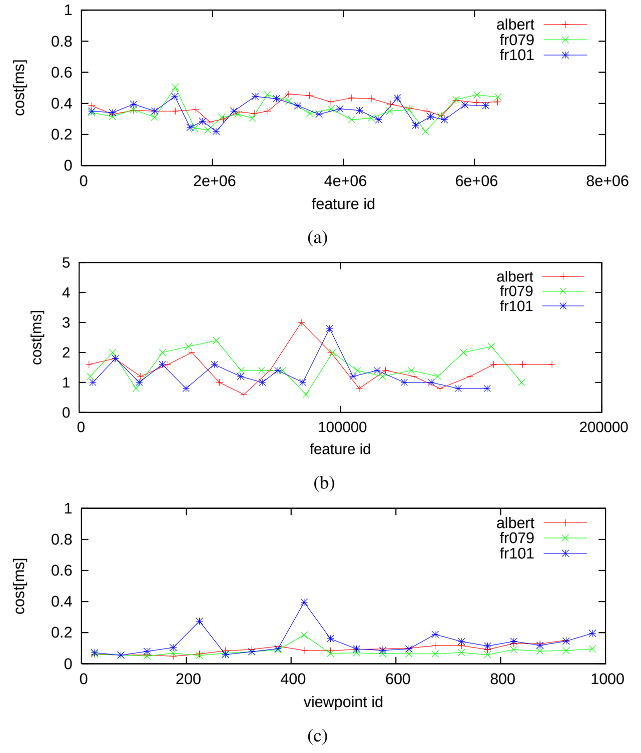


Fig. 4. Time costs of each process. (a) Database updating process (per feature). (b) Database retrieval process (per feature). (c) RANSAC map-matching process (per feature-hypothesis pair).

new features and new hypotheses are set to $|U^{new}| = 10$ and $|V^{new}| = 1,000$. The block size used for preemption is simply set as $B = |U^{new}|$. The parameters for LSH are set empirically to $K = 30$, $L = 20$.

We consider a typical scenario where R mapper robots are exploring R different buildings. There is no a priori knowledge on the buildings as well as on which buildings the individual robots are exploring. Each submap G_r $(1 \leq r \leq R)$ is initialized to an empty map at the beginning of the map-building task, then incrementally built by each mapper robot during the task. Every time a novel scan arrives, a set of appearance features are extracted from the scan points. Then, the grid maps as well as the appearance database are updated with the extracted features. Fig. 4(a) shows the time consumed by this updating process per feature for different size global maps for three different environments "albert", "fr079" and "fr101". It can be seen that the required time is independent of the map size and bounded by a constant. The direct implementation of a submap consumes around 2.1×10^7 cells in the 3D $xy\theta$ pose space and the global map consumes around 3.4×10^8 cells in total. Our system can translate the grid map into a UH hash table of size 1×10^7 bins. The size of UH table is less than 3% of the original grid map. The LSH table consumed by the LSH database is at most 7.2×10^6 bins in this experiment. It can be concluded that our structure is completely incremental and scales to large-size maps.

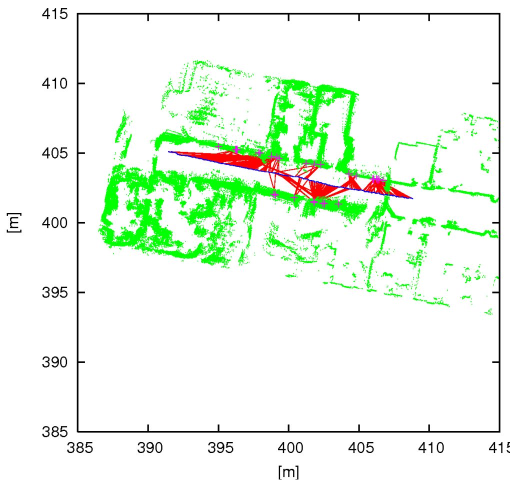


Fig. 5. Estimated trajectory. The blue curve indicates the estimated viewpoints (i.e. the top-ranked hypotheses). Each red line segment connects each hypothesis with the corresponding base feature point.

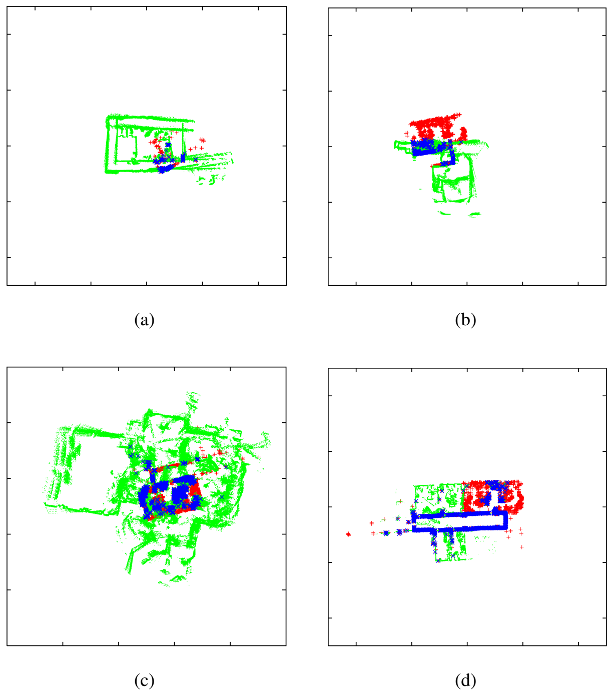


Fig. 6. Evolution of map-matching as we get more viewpoints from the target robot. (a),(b),(c),(d) The top-ranked hypotheses respectively at viewpoint #66, #292, #891 and #999. The green points are features in global maps. The blue and the red points respectively are inlier and outlier local features. While not yet correct in Figs. (a)(b)(c), the map matching is finally correct in Fig (d).

Fig. 5, 6 illustrate localization results. At the beginning of the localization task, the local map is initialized to an empty map. Every time a novel scan arrives, the local map is updated with the scan points. Then, a set of appearance features are extracted from the scan points. Then, the appearance database is retrieved using each feature as a key. Then, new hypotheses are generated based on the retrieval results. Fig. 4(b) shows that the time consumed by the retrieval process is actually bounded by a constant as explained in the previous section. Fig. 6(a),(b) and Fig. 6(c) show two typical cases where the map-matching is not yet successful. The first case is that the local map is yet not informative and as a result, all the matching results are not reliable. The second case is that most part of the local map is originated from *unknown* region which is not described in the global map and as a result, all the matching results are caused by false matches (e.g. the local map is matched with a wrong building). Fig. 6(d) shows that the matching finally becomes successful. This is because of that the local map now becomes informative and reliable so that much part of the map is originated from known region in the global map. Fig. 4(c) shows that the time consumed by the map-matching process is also bounded by a constant. Importantly, the time cost is independent of the size of map database.

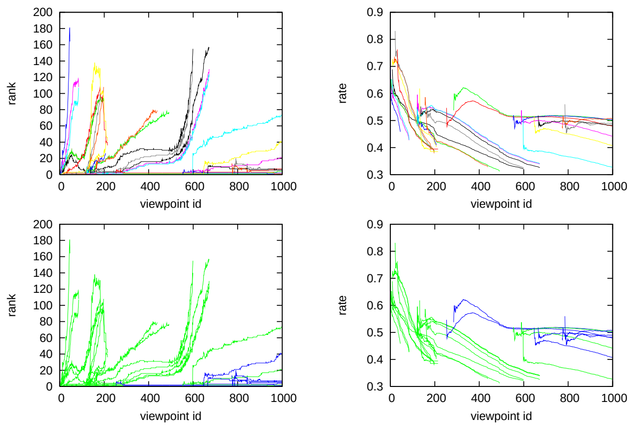
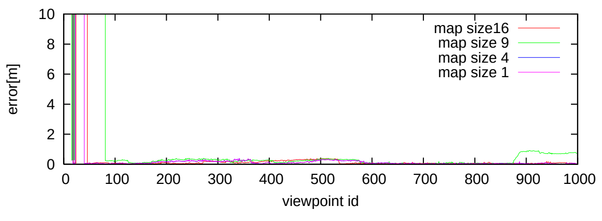
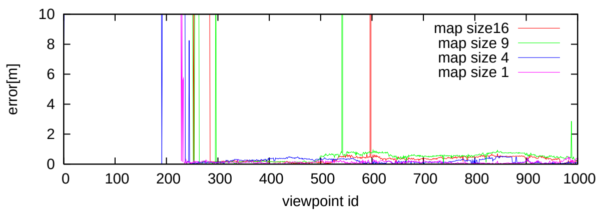


Fig. 7. Scoring results. The curves in the left and the right figures are the rank and the rate of individual hypotheses. For clarity, only those hypotheses that are top-ranked at least once are shown. The blue and the green curves in the bottom figures respectively indicate the correct and the wrong hypotheses.

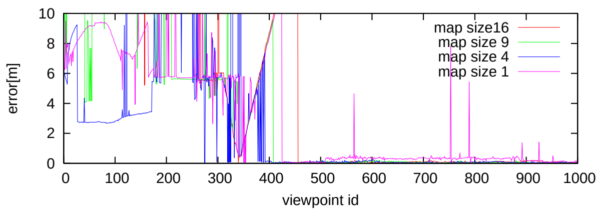
Fig. 7 illustrates scoring results. In the figure, ”rank” is a relative measure that indicates the height of score of one hypothesis against others. For the sake of clarity, only those hypotheses that are top-ranked (i.e. assigned rank 1) at least once are shown in the figure. The scoring by iRANSAC is performed in a preemptive manner similar with other rRANSAC schemes such as [25] and [35]. Good hypotheses supported by many data-points tend to be efficiently detected. Bad hypotheses contaminated by noises tend to be quickly weeded out. Once a good hypothesis is high-ranked it tends to stay in the high-rank group for a long period of time. A key difference from typical rRANSAC schemes is that the rate and the ranking of a hypothesis tend to decrease as the robot navigates. This is because of that old hypotheses tend to be inconsistent with new data-points due to the accumulation of errors in the odometry as well as in the local map. In consequence, old and new hypotheses are compared against each other in a preemptive manner and the top-ranked hypothesis supported by many recent data-points tends to be output as a best hypothesis at each viewpoint.



(a)



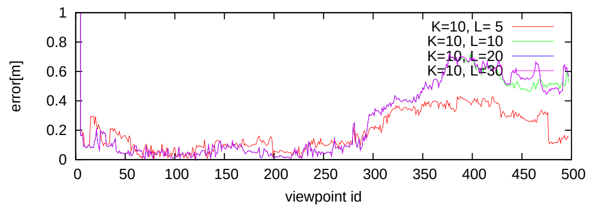
(b)



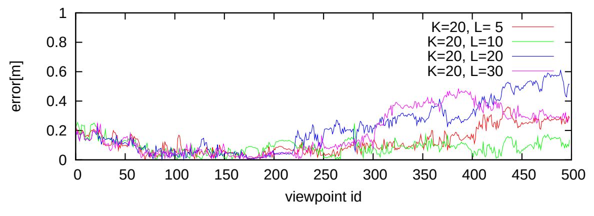
(c)

Fig. 8. Localization error results (y-axis) when the amount of measurements or iterations (x-axis) increases. (a) ”fr101”, (b) ”albert”, (c) ”fr079”. For each environment, the localization tasks are conducted for four different global maps respectively composed of 1, 4, 9 and 16 submaps.

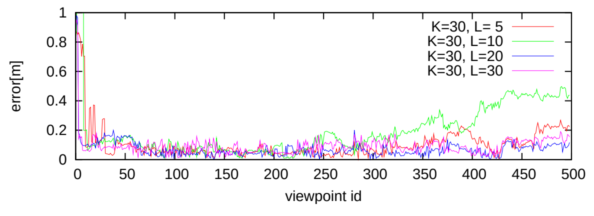
Fig. 8 reports the localization performance in several experiments. A series of localization tasks are conducted in 12 different scenarios, for three different local maps shown in Fig. 3 (a)-(c), and for four different size global maps respectively composed of 1, 4, 9 and 16 submaps. The target robot is assumed to be located in the same building as one of the mapper robots. A pair of non-overlapping measurement sequences are created from the same dataset with a procedure described in [16] and respectively assigned to the target robot and the mapper robot located in the same building. In the case of Fig. 8 (b) and (c), it can be seen that localization is not successful at the initial stage of the localization task. This is because of that the robot is initially located in unknown region that is not described in the global map. It can be seen for all the experiments that the localization errors finally become lower than 1 m. The global map size 16 corresponds to over 3.5×10^5 appearance features. In the Fig. 8, it can be seen that the localization task is successful even for 16 submaps environments. This is a quite large-size map compared with previously published works on fully incremental systems. It is noteworthy that the proposed incremental scheme is successful even for such a large-size map. We also conducted additional experiments to investigate the parameter sensitivity and the scalability of our scheme.



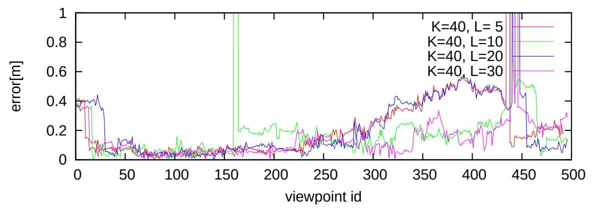
(a)



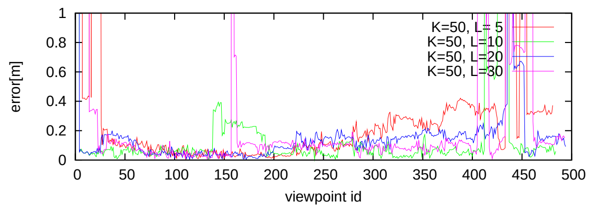
(b)



(c)



(d)



(e)

Fig. 9. Comparison of localization performance for different settings of K and L.

Fig. 9 summarizes the sensitivity against LSH parameters. It can be seen that our system is stable for a wide range of parameters. We also conducted experiments with much larger global maps using synthesized datasets, and found that our scheme is successful as much as 40 submaps (1.2×10^6 features) environments. From above results, it can be concluded that the proposed scheme is fully incremental and scales to large-size problems.

VI. Conclusions

The primary contribution of this paper is the development of a localization system that is completely incremental and scales to large-size environments. By using an LSH appearance database and a RANSAC map-matching scheme, we efficiently solve the real-time localization in large-size environments. The provided experimental results show the system can efficiently deal with increasing number of features in local and global maps which are incrementally arriving during the localization task. Because we combine the appearance-based and the position-based methods, our system would be effective even in large and dynamic environments. In future, we plan to further optimize the data structures as well as to apply our system to other platforms such as vision-guided mobile robots.

Acknowledgement

We are grateful to Dr. Alexandr Andoni for providing a useful code of LSH (Locality-Sensitive Hashing).

References

[1] Neira J., Tardos J.D., and Castellanos J.A. Linear time vehicle relocation in slam. *IEEE International Conference on Robotics and Automation*, 1:427– 433, 2003.

[2] Kin Ho and Paul Newman. Multiple map intersection detection using visual appearance. *3rd International Conference on Computational Intelligence, Robotics and Autonomous Systems*, 2005.

[3] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. *IEEE International Conference on Robotics and Automation (ICRA)*, 1999.

[4] Lars A. A. Andersson and Jonas Nygards. C-sam: Multi-robot slam using square root information smoothing. *IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 2798–2805, 2008.

[5] Rainer Kummerle, Rudolph Triebel, Patrick Pfaff, and Wolfram Burgard. Monte carlo localization in outdoor terrains using multilevel surface maps. *Journal of Field Robotics*, 25:346 – 359, 2008.

[6] Schindler G., Brown M., and Szeliski R. City-scale location recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1 – 7, 2007.

[7] Murillo A.C., Guerrero J.J., and Sagues C. Surf features for efficient robot localization with omnidirectional images. *IEEE International Conference on Robotics and Automation*, pages 3901–3907, 2007.

[8] J. Wang, H. Zha, and R. Cipolla. Coarse-to-fine vision-based localization by indexing scale-invariant features. *IEEE Transactions on Systems, Man, and Cybernetics*, 36:413–422, 2006.

[9] Lina M. Paz, Pdro Piniés, José Neira, and Juan D. Tardós. Global localization in slam in bilinear time. *Proc. 2005 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 655–661, 2005.

[10] Arturo Gil, Oscar Reinoso, Asuncion Vicente, Cesar Fernandez, and Luis Paya. Monte carlo localization using sift features. *Lecture Notes in Computer Science (LNCS)*, 1(3523):623–630, 2005.

[11] Nikos A, Vlassis, Bas Terwijn, and Ben J. A. Kroese. Auxiliary particle filter robot localization from high-dimensional sensor observations. *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2002.

[12] J. Wolf, W. Burgard, and H. Burkhardt. Robust vision-based localization by combining an image retrieval system with monte carlo localization. *Trans. IEEE Robotics*, 21(2):208–216, 2005.

[13] Jun Jie Foo and Ranjan Sinha. Pruning sift for scalable near-duplicate image matching. *Proc. Int. Conf. ACM*, pages 63–71, 2007.

[14] Andrew Johnson. *Spin-Images: A Representation for 3-D Surface Matching*. PhD thesis, Carnegie Mellon University, 1997.

[15] Mori G., Belongie S., and Malik J. Shape contexts enable efficient retrieval of similar shapes. *IEEE Computer Vision and Pattern Recognition*, pages 723–730, 2001.

[16] Tanaka K. and Eiji K. A scalable algorithm for monte carlo localization using an incremental e2lsh-database of high dimensional features. *IEEE Int. Conf. Robotics and Automation*, pages 2784–2791, 2008.

[17] Gionis A., Indyk P., and Motwani R. Similarity search in high dimensions via hashing. *Proceedings of the 25th Very Large Database (VLDB) Conference*, 1999.

[18] Yamauchi B. and Langley P. Place recognition in dynamic environments. *Journal of Robotic Systems*, 14, 1997.

[19] Tanaka K and Kondo E. Incremental ransac for online vehicle relocation in large dynamic environments. *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pages 1025–1030, 2006.

[20] Takeshi Ueda and Kanji Tanaka. On the scalability of robot localization using high-dimensional features. *IAPR International conference on pattern recognition*, 2008 (to appear).

[21] Andrew Howard and Nicholas Roy. The robotics data set repository (radish), 2003.

[22] F. Linaker and M. Ishikawa. Real-time appearance-based monte carlo localization. *Robotics and Autonomous Systems*, 54(3):205–220, 2006.

[23] B. Matei, Y. Shan, H.S. Sawhney, Y. Tan, R. Kumar, D. Huber, and M. Hebert. Rapid object indexing using locality sensitive hashing and joint 3d-signature space estimation. *Trans. IEEE PAMI*, 28(7):1111–1126, 2006.

[24] Goshen Liran and Shimshoni Ilan. Balanced exploration and exploitation model search for efficient epipolar geometry estimation. *Trans. IEEE PAMI*, 30:1230–1242, 2008.

[25] Chum O. and Matas J. Optimal randomized ransac. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:1472–1482, 2008.

[26] Ming Yang, Junsong Yuan, and Ying W. spatial selection for attentional visual tracking. *IEEE Computer Vision and Pattern Recognition*, 2007.

[27] Sivic J. and Zisserman A. Video google: a text retrieval approach to object matching in videos. *Ninth IEEE International Conference on Computer Vision*, pages 1470–1477, 2003.

[28] Yeh T., Lee J., and Darrell T. Adaptive vocabulary forests br dynamic indexing and category learning. *IEEE International Conference on Computer Vision*, 2007.

[29] Zhiwei Zhu, T. Oskiper, S. Samarasekera, R. Kumar, and H.S. Sawhney. Ten-fold improvement in visual odometry using landmark matching. *IEEE International Conference on Computer Vision*, 2007.

[30] Tuytelaars T., C. Schmid, and Leuven K.U.Leuven. Vector quantizing feature space with a regular lattice. *IEEE Computer Vision and Pattern Recognition*, 2007.

[31] Moosmann F., Nowak E., and Jurie F. Randomized clustering forests for image classification. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(9):1632–1646, 2008.

[32] 25 years of ransac. *Proc. IEEE Int. Workshop in conjunction with CVPR '06*, 2006.

[33] Ondrej Chum and Jiri Matas. Matching with prosac progressive sample consensus. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pages 220–226, 2005.

[34] O. Chum and J. Matas. Randomized ransac with $t_{d,d}$ test. *Proc. British Machine Vision Conference*, pages 448–457, 2002.

[35] David Nister. Preemptive ransac for live structure and motion estimation. *Machine Vision and Applications*, 16:321 – 329, 2005.

[36] Michael Montemerlo. *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association*. PhD thesis, Carnegie Mellon University, 2003.

[37] Juan Nieto, Tim Bailey, and Eduardo Nebot. Recursive scan-matching slam. *Robotics and Autonomous Systems*, 55:39–49, 2007.

[38] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality sensitive hashing scheme based on p-stable distributions. *Proc. Symposium on Computational Geometry*, pages 253–262, 2004.

[39] A. Andoni, M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. *Nearest Neighbor Methods in Learning and Vision: Theory and Practice*, 2006.