Vol.2010-CVIM-172 No.5
2010/5/27

Online Incremental Vision-based SLAM in Dynamic Cluttered Environment

Noppharit TONGPRASIT[†a)] Aram KAWEWONG[†b)]
Osamu HASEGAWA[††c)]

This paper presents a novel use of feature sharing in an appearance-based simultaneous localization and mapping (SLAM) system for robots. Feature sharing was inspired by man-made settings such as offices and houses, in which many similar objects are repeatedly shown in the same environment. With this concept, we can expect better performance with lower memory consumption. Combining this concept with Position Invariant Robust Features (PIRFs) [1], we can improve both accuracy and processing time. Our system is fully online and incremental. Our experiments were done on two well-known datasets, the City Centre dataset [2] and Lip6Indoor dataset [4]. Moreover, we tested our system on crowded university canteen at lunch time for more dynamic environment. The results showed that our system has outstanding accuracy and less processing time compared to FAB-MAP and fast and incremental bag of words which are considered the state-of-the-art offline and online appearance-based SLAM system, respectively.

変動要因が多い環境下でオンラインで稼働する画像を用いた自己位置同定手法

トンプラシット　ノッパリット[†a)]
カーウィーウォン　アラム[†b)]　長谷川　修[††c)]

本論文では、移動ロボットのための、完全にオンラインで稼働する、画像を用いた自己位置同定手法(Visual-based SLAM)を提案する。提案手法は、以下のようにまとめられる。（１）移動するロボットから観測される画像中に頻繁に現れるSURF特徴(PIRF特徴：Position Invariant Robust Features)を適応的に検出しIDをつける、（２）PIRF特徴とIDの対応関係についてまとめた辞書を作る、（３）各位置における入力画像（環境）を辞書に記されたIDで記述する、（４）辞書の更新・環境の記述・自己位置同定をオンラインで繰り返す。提案手法の有効性は、世界的に知られる"City Center"と"Lip6Indoor"データベースを用いて評価し、提案手法が従来手法より、認識率と処理速度の双方で約２～３倍優れることを確認した。さらに本研究では、世界で初めて、混雑する大学食堂の環境下で実験を行い、そうした極めて変動要因の多い（多くの人が動く）環境下でも、提案手法が辞書の作成と更新、自己位置同定をオンラインで確実に実行でき、高い自己位置同定精度が得られることを確認した。本研究は、画像を用いた自己位置同定の研究分野の新たな地平を切り拓くものと考える。

1. Introduction

Robots that can be used at home might soon step out of science fiction and into reality. Many researchers are attempting to create a complete humanoid robot. However, some functions are still inferior to those of even a newborn baby. One of these is the navigation system, which is essential in the next generation of home-use robots. Widely used navigation systems so far relied on laser scanners, which perform well in closed and unsophisticated environments but are not robust to noise, such as rough paths or crowded situations. The cost of these devices is high, making robots with such devices too expensive for ordinary people to be able to afford. On the other hand, camera technology is developing rapidly, and the cost is decreasing. For instance, a good cell phone with a built-in camera is currently priced as low as 100 dollars. Moreover, for a higher-level intelligent robot such as a humanoid, a camera serves as eyes. Humans do not need sonar to locate themselves but depend only on their eyes. Similarly, robots should be able to locate themselves by camera. This is where appearance-based simultaneous localization and mapping (SLAM) was first introduced.

In an online visual SLAM system, the size of the dictionary can be increased. Most online SLAM systems face a common problem: growth of the dictionary. The larger the environment is, the higher the memory required for storing word's describing feature. We used the fact that most robots are designed to be used in houses or in the city but not in the jungle. We can see that objects around us look alike. For instance, we can expect most of the doors to rooms in an office to be the same and the fire extinguishers to look alike everywhere. Instead of remembering each room's door, or in visual SLAM, every feature, the system can remember all look-alike features as one single feature, and thus slowing the growth of the dictionary and decreasing the computing time.

In this paper, we design our visual SLAM based on Position-Invariant Robust Features [1] (PIRFs) and experimented on the well-known City Centre and Lip6Indoor datasets collected by Cummins and Newman [2] and Angeli et al. [4], respectively. Our method can outperform these state-of-the-art approaches [2] [4] in terms of both time and accuracy. On the City Centre dataset, our system could perform at a 80% recall rate at precision 1. On the Lip6Indoor dataset, our system could perform at an 77% recall rate at precision 1, twice faster than [4] does.

Moreover, for the first time in this area of study, we took a further challenged on dynamic scene. We collected another dataset from crowded university canteen during lunch break where all scenes are full of moving object. This experiment was set up to test ability of system in environment that system was unable to train or test without any moving object in the scene. This kind of environment was also applied to public place such as hospital, train station or super market. The results showed that both state-of-the-arts [2] and [4] fail in localization; however, our system was still able to recover and correctly localize with more than 86% recall

† 東京工業大学大学院　総合理工研究科　知能システム科学専攻
†† 東京工業大学　像情報工学研究所
a) E-mail: tongprasit.n.aa@m.titech.ac.jp
b) E-mail: kawewong.a.aa@m.titech.ac.jp
c) E-mail: hasegawa.o.aa@m.titech.ac.jp

at precision 1.

All the results showed that our proposed method can perform in real time even under MATLAB environment. Each image requires less than 1 second to search and correctly localize. We strongly believe that our method can perform faster in real implementation.

2. Related Work

As mentioned in the introduction, when SLAM was introduced, most systems relied on laser scanners. However, the difficulty of loop closure in metric SLAM [5] [6] seems to remain unresolved. In contrast, appearance-based SLAM recently became popular among robotics researchers because of the developments in technology of vision. Eade and Drummond [7] attempted to use the advantages of cameras with metric SLAM, but their method did not support large-scale environments, e.g., the City Centre dataset.

A few years ago, several visual SLAM systems were introduced. Experiments showed that the recall rate was too low for real applications. SLAM is developing in two directions: offline and online SLAM. Offline and online refer to the system's dictionary. Offline SLAM's dictionary is fixed and cannot be increased, whereas online SLAM's dictionary can be increased if necessary. Researchers must decide whether their system should go online or stay offline. An offline SLAM can handle calculations probabilistically (as shown by Cummins and Newman [2]) because the dictionary is closed. Thus, offline SLAM can perform quickly because the dictionary is fixed, but it must be very good. It is however difficult to judge which dictionary is the best because we cannot predict which environment the robot will be in. As a result, a robot will generally be more suited for a certain terrain and will require a different dictionary when placed in another environment. Later, in the following year, Cummins and Newman [3] introduced an improvement in model update for long run but they did not improve accuracy.

In contrast, online SLAM's dictionary can be increased. Thus, if there is no matching word, the system will register a new word in the dictionary automatically. However, two factors hinder the performance of online SLAM. First, the dictionary's size is incremental. The calculation seems to be more complicated, since the size of dictionary is not fixed. Angeli et al. [4] proposed fast and incremental bag of words(BOWs) with a four-step calculation. Their proposed method was not robust to dynamic scenes such as crowded or outdoor environments. The other factor is an increase in computational time. Fast and incremental BOWs [4] faced an immense computational time problem after processing because the system described in [4] was unable to decide which features were worth remembering. Some information is useless and this affected system performance.

Kawewong et al. [1] suggested that features showing motion can be neglected. They first proposed the method Position-Invariant Robust Features (PIRFs) [1]. A PIRF is extracted by finding the average of well-known SIFT local features [8] projected from the previous image. Their experiments showed remarkably good localization in a dynamic environment. Later, Noppharit et al. [9] proposed incremental area localization by PIRFs, which improved a visual SLAM system's recall rate to 60%. Based on PIRF's remarkably good performance, we applied a few of its concepts in our method.

Using local features alone, like in [4], the system might slow down and eventually stop if the user leaves it running long enough. The problem is the growth in dictionary size. In this paper, we also proposed a memory-sharing method to suppress this growth by dividing matching process into two parts. Instead of SIFT, we applied a speeded up robust feature (SURF) local feature with the PIRF concept to reduce computational expense. Our method showed remarkable recall rate, precision, and calculation time in localization.

3. SURF-based PIRF

In [1], Kawewong et al. suggested tracking local features, such as SIFTs, that appear continuously during a specified sliding window. The size of this sliding window varies inversely with the speed of the vehicle. If the sliding window is too large, the number of local features will be small. In contrast, if we designate a small sliding window, the number of local features will become too large, which negatively affects the matching process and computational time. To eliminate unpredictable factors, we proposed a new PIRF structure. We control only the number of local features for each incoming image and let the system adjust the size of the sliding window automatically. As Fig. 1 shows, the system needs to control the number of local features for each place. If there are too many features, the system will extend the sliding window automatically. Conversely, if an extended sliding window removes many features, the system will shrink back to the former sliding window automatically. The connection list stores only the connection from each location to the previous one, so the system does not need to calculate every match from the beginning again.

Kawewong et al. chose SIFT as a descriptor in [1]. However, SURF [10], which was proposed by Herbert et al., is a good alternative. Depending on the dimensional size, SURF features might contain less descriptive power than SIFT; however, their computational cost is much lower than that of SIFT. In this study, we used SURF as the basic local feature. Moreover, we also neglected the average feature calculation of Kawewong et al. because it does not affect the matching process much.

4. Matching Process

Here, we divide matching into two parts: scoring matching and model-generating matching; these will be described in Section VI. First, we assume that the current location at time t, Lt, is a previously visited place. Dict-1 is a dictionary generated from previous location Lt-1. If the system found all features of Lt in Dict-1, Lt would match Lt-1, because neighboring locations usually resemble the current location most closely. To prevent this, the system searches from Dict-offset instead. Offset is a number of locations near Lt that should not appear in this matching process. We use Euclidean distance ratio to match each entries with words in Dict-offset. The search result will appear in the array Appeart, each member of Appeart that represents a feature matching one in Dict-offset. If a feature cannot be paired with any from Dict-offset, then that entry will be left as 0, as shown in Fig. 2. Appeart will be used for score calculation in the next section.

5. Score Calculation

Vol.2010-CVIM-172 No.5
2010/5/27

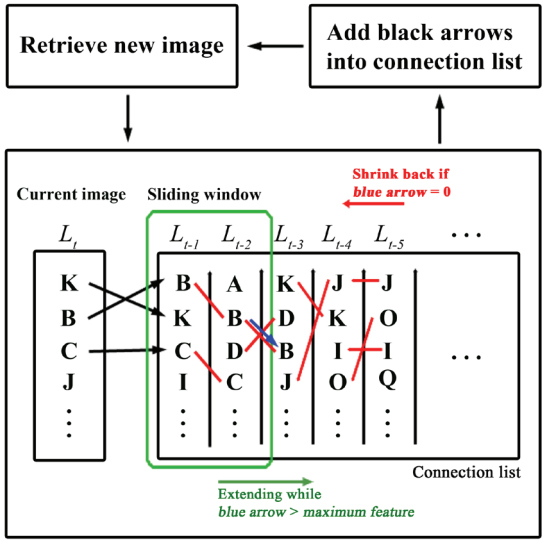

Fig. 1. Local feature extraction based on PIRFs. After image retrieval, the system matches the features of location Lt with those of other locations by tracking the number of connections in the list. In this figure, each letter represents a local feature. For the current state, feature B is the only one that locations Lt, Lt-1, and Lt-2 have in common. This feature is stable and worth remembering. The system extends the sliding window until the number of matching features (blue arrow) between Lt and the last location in the sliding window is below a threshold. If the sliding window cannot find any matching features, the system shrinks the sliding window back to its former state. After matching, the system will update the list by pushing matching features (black arrow) into the list; it is then ready for a new image.

After obtaining $Appear_t$, the system proceeds to the score calculation process. Angeli *et al.* [4] used modified *term frequency-inverted document frequency* (*tf-idf*) [11] for scoring. *Tf-idf* is a good method for weighing every feature with the number of occurrences. It can perform well with a large number of candidate features, like SIFT. In one image, SIFT can extract more than 2000–3000 features, whereas SURF can extract only 200–300. Moreover, the number of selected features from PIRF will gradually decrease to less than 100. Thus, *tf-idf* is not applicable for SURF-based PIRF. We did not use *tf-idf*; instead, we weighted the score according to the number of known features from $Dic_{t\text{-}offset}$, because the higher the number of matching features found, the more likely it is that the current image is a known place.

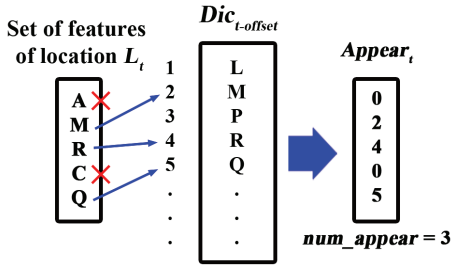

Fig. 2. Matching process diagram. In current location Lt, the system find matching features in Dict-offset and put the index numbers into Appeart. Features that cannot be matched with any in Dict-offset will remain 0.

The number of non-zero entries of $Appear_t$ is *num_appear*. That is, *num_appear* is the number of features of L_t matching $Dic_{t\text{-}offset}$. The similarity score can be calculated using (1). n_m is the number of features that coappear in both visited locations at time m, L_m and current location L_t. Multiplying the number of coappearing features in *num_appear* provides more priority by the location L_t as a known place. *Num_appear* does not affect the score of individual locations, but rather the overall score.

$$s_m = n_m * num\_appear \tag{1}$$

Due to the stability of PIRFs, features appearing in location L_m should also appear in neighboring locations such as L_{m-2}, L_{m-1}, L_{m+1}, and L_{m+2}. The score of nearby locations should be approximately less than or equal to s_m. For example, s_m has the highest score, but if s_{m-1} and s_{m+1} are 0, L_m should not be the same location as L_t. To filter out this kind of score, the system needs to consider nearby scores as well. In (2), b_m represents a second-state score of location L_m after considering neighboring scores. A transition probability generated from the Gaussian distance between times m and i, $p_T(m,i)$, was applied in (2) to give more weight to a closer place. w represents the number of neighboring locations we considered in calculating the score.

$$b_m = \sum_{i=m-w}^{i=m+w} (s_i \cdot p_T(m,i)) \tag{2}$$

After the system obtains all second-state scores of L_m to $L_{t\text{-}offset}$, it can clearly decide whether to accept L_t as a known place or reject it as a new place. For better score evaluation, we normalize the second-state scores into a specific range. We decided to use a simple integration estimation of second-state scores, as shown in (3). The normalizer n is varied according to the vehicle's velocity and the allowed maximum number of features during PIRF

extraction. The neighboring score reappears in the normalization process because we would like to emphasize the significance of neighboring locations and smoothen the score curves.

$$b\_norm_m = \frac{\sum_{i=m-w}^{i=m+w} b_i}{2w \cdot n} a \quad (3)$$

6. New Place or Known Place

After the system normalizes the second state, it decides whether current location Lt is new or known using the maximum value of b_norm. If the maximum of b_norm is greater than an acceptance threshold, the system will decide that location Lt as a known place.

6.1 New Place

If the maximum of *b_norm* is less than the threshold, system will decide that location L_t is a new place. In the section regarding matching, we mentioned that matching is separated into two parts: scoring matching and model-generating matching. In scoring matching, the system matched incoming features of L_t with $Dic_{t\text{-}offset}$. Thus, L_t has not been matched with current dictionary Dic_{t-1} yet. The system will match features that are unmatched with any feature in $Dic_{t\text{-}offset}$ with the current dictionary Dic_{t-1} and update $Appear_t$. As in matching for scoring, unmatched features will remain 0.

Currently, the system has a set of $Appear_t$; however, unknown features must be registered in the dictionary. We separate the features of L_t into two groups, *known_feat* and *unknown_feat*. The system registers *unknown_feat* in the dictionary and returns Dic_t, as in (4).

$$Dic_t \leftarrow Dic_{t-1} \cup unknown\_feat \quad (4)$$

At this point, the system will create a model of L_t by using index numbers that are not 0 in $Appear_t$ and the recently registered index feature from Dic_t.

6.2 Known Place

When the maximum of *b_norm* is greater than the acceptance threshold, the system decides that location L_t is a known place. The system picks candidate locations based on the greatest *b_norm* value and decide which candidate location is the same as L_t by selecting the highest similarity score. Using the similarity score will limit the confidence of the system, because the system relies on the neighboring score during second-state score calculation. Otherwise, the system might select some other neighboring location not matching ground truth.

Suppose that location L_t is the same place as a location at time k, L_k ($k < t$). The next step is to update the model. Instead of directly updating the model of L_k, the system generates a new model of L_t based on the model of L_k by updating the model of L_k with new features not originally included.

7. The Dictionary's Timeline Buffer

In Section IV, we mentioned the dictionary at time *t-offset*, $Dic_{t\text{-}offset}$. In fact, the system does not store all dictionaries all the time because of memory limitations. We create another buffer called *forbidden*, which is a simple first-in-first-out *offset*-sized stack. Each stack remembers the list of indexes of newly registered features. When processing scoring matching, the system simply excludes all features' indexes in Dic_{t-1} from searching. On the other word, system excludes the words added during time *t-offset+1* to *t-1* from being matched in the scoring matching process.

After updating the model, the system will pop out the first stack and push the newly added features' indexes in at the last stack. By this method, we can convert the current dictionary Dic_{t-1} into $Dic_{t\text{-}offset}$ for score calculation.

8. Experiment and Result

We conducted the experiments using the well-known datasets, the City Centre and Lip6Indoor datasets collected by Cummins and Newman [2] and Angeli *et al.* [4]. These datasets are available on their web sites. We also collected more challenging data and tested our system on a dynamic scene. The maximum number of local features allowed was set to 75 per location. Although the fast and incremental BOWs [4] code is unavailable, fortunately Angeli *et al.* kindly tested these datasets and provided us with the results. However, the results were simply the final answer, so we were unable to include fast and incremental BOWs' precision and recall curves in the graphs.

8.1 Experiment I: City Centre Dataset

This experiment was intended to demonstrate the performance of our system compared to those of [2] and [4] for an outdoor dataset. The City Centre dataset was collected by Cummins and Newman. The images were taken by stereo cameras at a frequency of one image every 1.5 meters. A total of 1237 locations were photographed for a total of 2474 images (left and right for each location).

Fig. 3. Result of City Centre dataset analysis. Yellow dots represent the vehicle's trajectory. Red dots represent the location with loop-closure detection. Left figure shows the aerial result of our system. Right figure shows the aerial result of FAB-MAP [2].

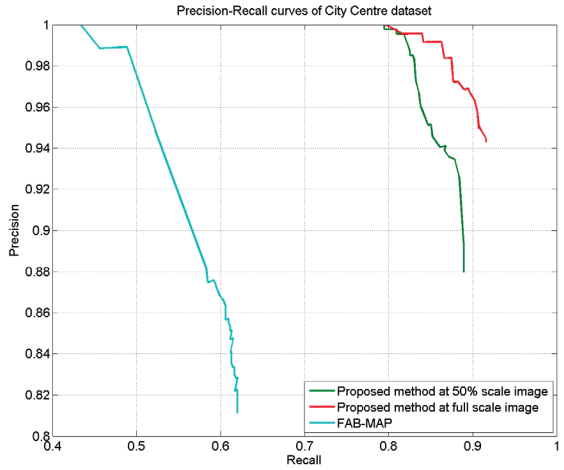

Fig. 4. Precision and recall curves for the City Centre dataset.

Fig. 3 presents an aerial image of the loop-closure result for the City Centre dataset. Fig. 4 shows the precision and recall curves compared with those of FAB-MAP. According to Fig. 4,

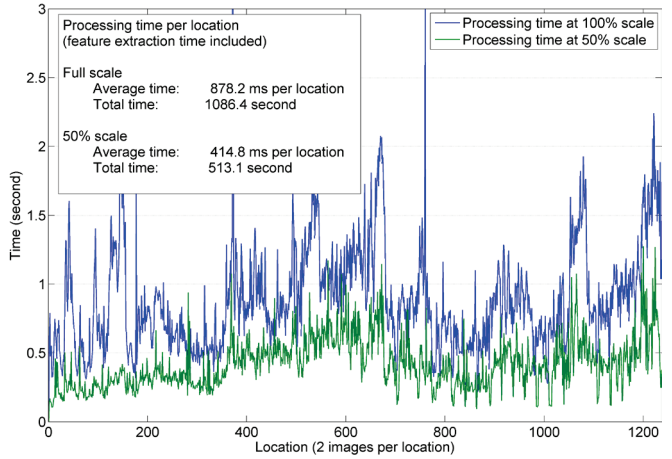

Fig. 5. Calculation time of City Centre Dataset. The system could finish the task 2 times faster at 50% scale, whereas the precision and recall remained almost the same.

Table I. The result of City Centre dataset at full scale

Method	Recall	Precision	False Positive	Total Time (sec)
Proposed method	80.03 %	100 %	0	1086.4*
FAB-MAP [2]	43.32 %	100 %	0	577
Fast and incremental BOWs [4]	23.89 %	97.76 %	2	7200+

* Run on MATLAB.

even if we compressed the input images to 50% scale, the scale does not affect the result. However, the system can finish the task much faster. Fig. 5 shows the calculation times at 100% scale and 50% scale. From Fig. 5, our system used less than 1.5 seconds at the last location. That is, our system can obviously run in real time (processing time is less than image retrieval time) even our method was run on MATLAB. Table I compares the results of our proposed system with those of other well-known methods. Our method could recall 79.68% of all locations at precision 1, which is almost twice as much as the state-of-the-art FAB-MAP could.

8.2 Experiment II: Lip6Indoor dataset

The purpose of this experiment was to evaluate the performance of our system with an indoor dataset. This dataset was collected from corridor in the building without any moving object by Angeli *et al.* [4]. Unlike the City Centre dataset, the input images were taken by a

Table II. The result of Lip6Indoor dataset at full scale

Method	Recall	Precision	False Positive	Total Time (sec)
Proposed method	77.73 %	99.42 %	1	32.84*
FAB-MAP [2]	23.64 %	100 %	0	187.16
Fast and incremental BOWs [4]	68 %	100 %	0	99

* Run on MATLAB.

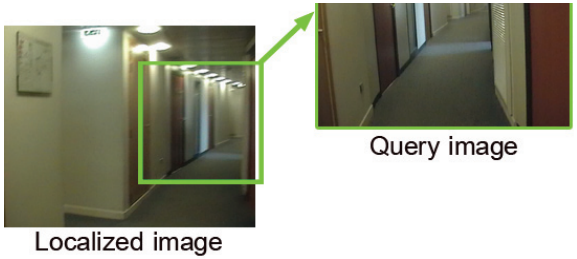

Fig. 6. False-positive localized location. Upper-right: query image; lower-left: image localized by the system.

conventional lens. This dataset was taken from the corridor of a building. A total of 318 images were collected at 1 frame per sec.

An indoor scene contains less information than an outdoor dataset. We decided to change the dictionary in FAB-MAP for greatest efficiency. Because FAB-MAP [2] is an offline loop-closure, we used the indoor vocabulary provided by Cummins and Newman in FAB-MAP.

Table II shows the results for the Lip6Indoor dataset. In this dataset, our system detected one false-positive location during the experiment. Fig. 6 shows the false-positive pair; the query image is on the right, and the image the system localized is on the left. The locations of these two images are almost the same, but they failed to achieve ground truth. From Table II, our method (84.6 ms per image) can complete the task three times faster than [4] (255.2 ms per image) and six times faster than [2] (482.4 ms per image) because we decrease the allowed maximum feature number, which affected the system's speed.

8.3 Experiment III: Crowded university canteen data

According to good results on 2 competing datasets, we decide to take another step of challenge in this area of study by focusing more on environment which contains more moving objects. We collected data from a university canteen during lunchtime. When we collected the data, to avoid robotic maneuvering problems in a crowded situation, we installed an omni-directional lens camera on the trolley and slowly recorded the data while moving around the crowded canteen. This dataset represents the environment where robot does not have a

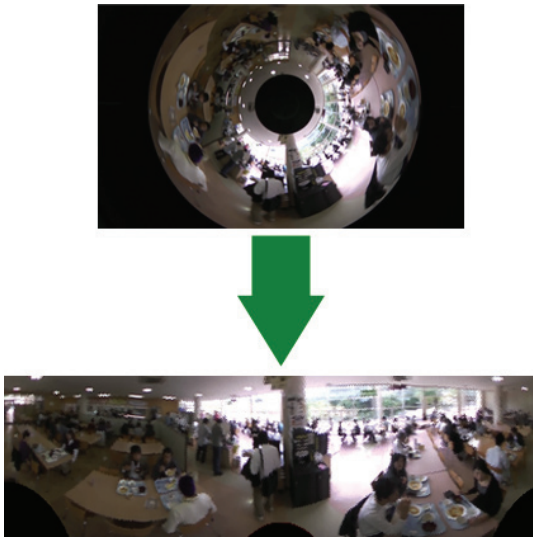

Fig. 7. Upper image is an example of collected data. Lower image is the unwrapped image.

chance to train under ideal condition where no moving object in the scene e.g. public places, train stations, crowded street sidewalk or department stores.

Fig. 7 shows an example of the collected data. The images were collected at two frames per second. The input image size is 270 × 480. We imaged 692 locations, for a total of 692 images. Figs. 8 and 9 show the vehicle's trajectory and the precision-recall graph, respectively. Refer to the Appendix for examples of matching results. Table III illustrates the result for each method. In this dynamic environment, our system showed outstanding results compared to FAB-MAP and fast and incremental BOWs in both accuracy and performance.

9. Discussion and Future Work

In the Experiment section, the overall results have already proven our system's performance. Our method can correctly localize better than other baselines. Our results clearly show that FAB-MAP did poorly on experiments 2 and 3. One major reason might be related to the dictionary. Although we used a dictionary for indoor environments on indoor datasets, FAB-MAP cannot achieve a high recall rate. This might raise questions about the definition of a good dictionary. How can we know which dictionary to use? This seems to be an open question, because we cannot know the quality of a dictionary unless it is tried and tested, which seems impractical for real use.

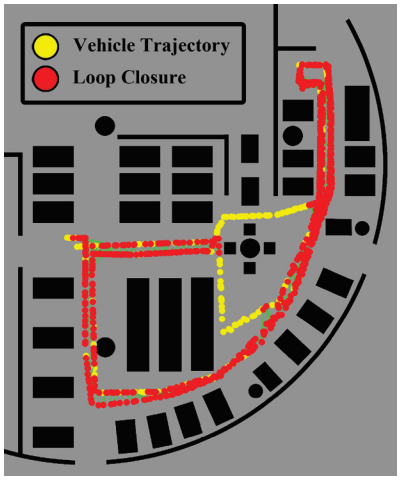

Fig. 8. Vehicle's trajectory. Yellow dots represent the vehicle's trajectory. Red dots

Table III. The Results for crowded university canteen dataset

Method	Recall	Precision	False Positive	Total Time (sec)
Proposed method	86.65 %	100 %	0	264.12*
FAB-MAP [2]	17.80 %	100 %	0	577
Fast and incremental BOWs [4]	1.01 %	100 %	0	2807

* Run on MATLAB.

On the other hand, fast and incremental BOWs [4] performs well with the Lip6Indoor dataset, but it failed to localize in the City Centre and university canteen datasets. This is because fast and incremental BOWs [4] remembered all features that had appeared. Some features might focus on moving people or cars. From the results for the City Centre dataset, more than two hours of calculation for fast and incremental BOWs [4] is high compared to other methods, because fast and incremental BOWs did remember all the local features. The number of features in the dictionary directly affects the computational cost. The bigger the dictionary is, the slower the system becomes.

Unlike [2] and [3], our system does not require a preliminarily generated dictionary, and because of the modified PIRF concept, we can control the number of features per location, and those features are stable and worth remembering and cause better result than [4]. Dividing the matching process into two parts, scoring matching and model-generating matching, can avoid unnecessary growth of the dictionary.

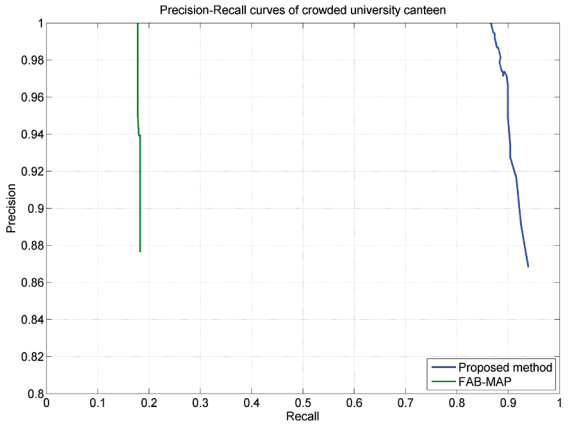

Fig. 9. Precision and recall curves for crowded university canteen.

In experiment 1, we also showed the precision and recall curves for the City Centre dataset at 50% scale. Although we compressed the images to 50%, the result did not show any significant effect on the accuracy. Moreover, the task can complete at two times faster rate than at full scale, because all selected features are perfect for the localization system.

All experiments described in this paper were programmed and computed in MATLAB on an Intel Xeon 2.66 GHz CPU. We believe that our system can complete the tasks faster when implemented in the C language. Despite our good results, our scoring system is not standard and must be improved in the future.

10. Conclusion

This paper proposed an alternative method for appearance-based SLAM. Our method is fully incremental and online. With a memory-sharing technique and the PIRF concept, our system showed outstanding results in both accuracy and computational cost. The experimental results showed that our method can efficiently handle all processes in real time. We strongly believe that our work would make appearance-based SLAM more practical in real applications.

Acknowledgement

This work reported in this paper was funded by an Industrial Research Grant Program received in 2009 from the New Energy and Industrial Technology Development Organization (NEDO). The authors gratefully acknowledge Oxford Robotics Research Group and ENSTA Electronics and Computer Engineering Laboratory for their provided databases and experiment results.

References

1) A. Kawewong, S. Tangruamsub, and O. Hasegawa, “Wide-Baseline Visible Features for Highly Dynamic Scene Recognition,” in Proc. Int'l. Conf. Computer Analysis of Images and Patterns (CAIP), 2009, vol. 5702, pp. 723–731.
2) M. Cummins and P. Newman, “FAB-MAP: Probabilistic Localization and Mapping in the space of Appearance.” in Int'l. Jour. Robotics Research, 2008, 27(6), pp. 647–665.
3) M. Cummins, and P. Newman, “Highly Scalable Appearance-Only SLAM - FAB-MAP 2.0,” Proc. Robotics: Sciences and Systems (RSS), 2009.
4) A. Angeli, D. Filliat, S. Doncieux, and J. A. Meyer, “Fast and Incremental Method for Loop-Closure Detection Using Bags of Visual Words,” IEEE Trans. Robotics, 2008, 24(5), pp. 1027–1037.
5) D. Filliat and J. -A. Meyer, “Map-based Navigation in Mobile Robots-I. A Review of Localisation Strategies,” Cognitive Systems Research, vol.4, no. 4, 2003, pp. 243–282.
6) J. -A. Meyer and D. Filliat, “Map-based Navigation in Mobile Robots-III. A Review of Map-learning and Path-planning Strategies,” Cognitive Systems Research, vol. 4, no. 4, 2003, pp. 283–379.
7) E. Eade and T. Drummond, “Unified Loop Closing and Recovery for Real Time Monocular SLAM,” British Machine Vision Conference (BMVC), 2008.
8) D. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” Int'l. Jour. Computer Vision (IJCV), 60(2), 2004, pp. 91–110.
9) T. Noppharit, A. Kawewong, and O. Hasegawa, “Data Partitioning Technique for Online and Incremental Visual SLAM,” in Proc. Int'l Conf. on Neural Information Processing (ICONIP), 2009, vol. 5863, pp. 769–777.
10) H. Bay, A. Ess, T. Tuytelaars, and L. Gool, “SURF: Speeded Up Robust Features,” Computer Vision and Image Understanding (CVIU), 110(3), 2009, pp. 346–359.
11) J.Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," inProc. IEEE Int'l Conf. Comput. Vision (ICCV), 2003, vol. 2, pp. 1470–1477.M. Cummins and P. Newman, “FAB-MAP: Probabilistic Localization and Mapping in the space of Appearance.” in Int'l. Jour. Robotics Research, 2008, 27(6), pp. 647–665.

Appendix

Note: Some correct matching examples from the crowded university canteen. Query images are on the left, and images on the right are those recognized by our system. These images were taken by an omni-directional lens. For better understanding, we unwrapped these images into panorama view. The center of the images is the front of vehicle, and the left and right sides are the backside of the vehicle. Regardless of the direction of vehicle, the system can detect loop-closure without fail.

* Some images might look different because the camera directions differ.