2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

Indexing Visual Features: Real-Time Loop Closure Detection Using a Tree Structure

Yang Liu and Hong Zhang

Abstract— **We propose a simple and effective method for visual loop closure detection in appearance-based robot SLAM. Unlike the Bag-of-Words (BoW hereafter) approach in most existing work of the problem, our method uses direct feature matching to detect loop closures and therefore avoid the perceptual aliasing problem caused by the vector quantization process of BoW. We show that a tree structure can be efficient in online loop closure detection. In our method, a KD-tree is built over all the key frame features and an indexing table is kept for retrieving relevant key frames. Due to the efficiency of the tree-based feature matching, loop closure detection can be achieved in real-time. To investigate the scalability of the method, we also apply the scale dependent feature selection in our method and show that the run time can be reduced significantly at the expense of sacrificing the performance to some extent. The proposed method is validated on an indoor SLAM dataset with 7,420 images.**

I. INTRODUCTION

Loop closure detection requires a mobile robot to recognize previous places accurately and correctly when they are revisited. It is considered an essential problem in simultaneous localization and mapping (SLAM). Correct loop closure detection can help the robot reduce the uncertainty associated with the system states and amend the accumulated errors during the mapping process, while incorrect claim of loop closure can be catastrophic as it will either introduce redundancy or imply incorrect update to the map, which can finally lead to mapping errors. Loop closure detecion is now attracting increasing attention in SLAM research.

Due to the popularity of visual sensors (such as monocular or panoramic vision), people have recently introduced appearance-based method to SLAM community. Appearance-based SLAM maintains a topological representation of the environment where locations are described by images taken by the robot camera in a continuous manner. The topological relationship among distinct places is kept during the mapping process to help with the robot localization. With such a map representation, loop closure detection can be achieved by comparing and matching images.

Researchers in computer vision community borrowed the idea of *indexing* and BoW from text retrieval field [1] and applied it extensively to image matching and classification [2]. Since the basic issue in visual loop closure detection is image comparison and matching, robotics people have also introduced the BoW method to appearance-based SLAM. Although BoW has gained considerable success and actually become the most popular technique in visual loop closure detection, it suffers from the perceptual aliasing problem [5] caused by the clustering which is used to generate the visual words. Perceptual aliasing causes both a low recall due to missed loop closures and a high cost in geometrical verification for loop closure candidates that need to be processed.

Directly matching raw features extracted from two images represents an reasonable similarity measurement and can be used for visual loop closure detection [8]. In other words, it is possible to detect loop closures by using the features themselves rather than their vector quantized representation [7], [9]. The problem is however, feature matching can be time-consuming if linear search is used. In fact, direct feature matching via linear search can soon become intractable with the increasing number of images to be processed and prevent SLAM from real-time implementation. In contrast, a tree structure is an efficient data structure for feature matching [3] and can provide online processing in detecting loop closures due to its efficiency. In our two-step algorithm, a KD-tree is built over all the key frame features and a tree search is performed for the features extracted from the query image. The following feature matching step in image level is then applied to confirm the correct loop closure candidates. Inspired by the indexing technique used in BoW, we also maintain an indexing table for fast retrieval of the loop closure candidates. Our method, referred to as indexing visual features, shows superior performance to BoW on a moderate size dataset in terms of both recall and run time.

The rest of the paper is organized as follows: Section II briefly summarizes the recent related work using both BoW and feature matching. We discuss our proposed method in detail in Section III. Experimental results from a moderate size indoor dataset are provided in Section IV. Finally, we end the paper with conclusion and discussion in Section V.

II. RELATED WORK

Visual loop closure detection using BoW is now extensively studied in robotics research. Newman *et al.* developed a SLAM system in his early work [4] with multiple sensors. A geometric 3D map was built using a laser range finder with the aid of visual sensor to detect loop closures based on appearance recognition. Angeli *et al.* has done a series of work in appearance-based SLAM. The authors incorporated Bayesian framework to BoW representation to compute the likelihood of loop closure occurance [11]. The performance could be improved since the information from previous observation was also taken into account. The work

This work was supported in part by NSERC

Yang Liu and Hong Zhang are with the Department of Computing Science, University of Alberta, Edmonton, AB, Canada {liu17, hzhang}@cs.ualberta.ca

was expanded in [12] with the other local feature of the color histogram besides SIFT. Based on their loop closure detection work, a topological SLAM system was built in [13] and metric information from robot odometry was integrated in [14] to obtain a globally consistent environment model and achieve global localization task.

Cummins and Newman set up a milestone for appearance-based SLAM called FAB-MAP [5]. The authors used a Chow-Liu tree to capture the co-occurrence statistics of the visual words and proposed a method for explicitly calculating the normalizing term in the recursive Bayes estimation for location likelihood. The probabilistic framework was a success in challenging outdoor datasets in detecting loop closures. This model was modified in FAB-MAP 2.0 [6] in conjunction with a randomized KD-forest used in the clustering process for generating visual words for the system to deal with large outdoor environment.

Most recently, people have used direct feature matching in loop closure detection to avoid the inherent perceptual aliasing problem (as discussed in Section I) in the above methods[1]. The authors in [7] proposed a new kind of local feature named position-invariant robust feature (or PIRF) to handle the problem. PIRF is in fact the stable local features that can be tracked in several continuous frames. Similarly, Zhang used scale dependent feature selection [9] to discard most SIFT features extracted at fine scales. The remaining coarse-scale features are more likely to be matched compared with those discarded [10], yet the number is much less. The purpose of both methods is to select a small number of informative features that can better describe the scene.

III. FEATURE MATCHING USING KD-TREE

Tree structures such as KD-tree and randomized KD-forest have been widely used in vision community for efficient nearest neighbor search. Hierarchical k-means [15] is an example of expanding the work in [1] to improve the scalability of the method. Randomized KD-forest is shown to have better performance than a single KD-tree [3], [16], [17]. Compared with the existing work on visual loop closure detection where KD-tree is used in clustering, our method uses KD-tree to detect loop closures via feature matching and no vector quantization is used. The idea is inspired by several observations.

We first notice that KD-tree not only reduces the search complexity from linear to logarithmic, but also compares one dimension of the high-dimensional features each time and therefore avoids the distance computations, the most time-consuming part in finding the correct nearest neighbor for a query feature with linear search. The previous work such as [6] used KD-tree (or radomized KD-forest) to assign each feature to its closest cluster center during the clustering process. This could make the clustering possible with a large amount of features to generate tens of thousands of visual words. However, the perceptual aliasing problem may deteriorate by erroneously classifying the features to their corresponding cluster centers, although we have no way to validate how significantly this approximate k-means clustering [16] may impact the performance of loop closure detection since standard k-means over too large dataset is intractable and no comparison can be made then. Nonetheless, KD-tree can still be a strong and efficient tool for approximate nearest neighbor search, which is an essential part of our method. In fact, people have shown the possibility of using KD-tree or randomized KD-forest in matching high-dimensional image descriptors [3], [17].

The second observation is that the distance ratio method [18] is an effective way for verifying putative matches. By using the distance ratio test, a correct match requires the ratio between the distances of the closest and second closest neighbor to the query feature to be below some given threshold. This implies that we can have a way of handling incorrect nearest neighbors returned by a KD-tree. In our work, we focus more on the applicability of the tree structure in feature matching by using distance ratio technique.

Some other phenomena that can make KD-tree suitable to our work will be discussed later as these observations can be explained clearly together with the proposed algorithm.

A. Key Frame Selection and Construction of KD-tree

In order to study the performance of loop closure detection, we first need to select a set of key frames. Key frames are representative images taken from distinct places of the environment. Usually there is no need to keep all the captured images since many consecutive ones look similar and a subset of sampled images are enough. We use the key frame extraction strategy proposed in [8] with some modifications to select key frames. Particularly, the first image captured is always considered as the first key frame and two consecutive key frames should either have little overlap or be "far apart" enough spatially, whichever occurs first:

$$overlap = \frac{2N}{N_i + N_t} \tag{1}$$

$$distance = |ID_i - ID_t| \tag{2}$$

where N_i and N_t are the numbers of scale-invariant features (we use SIFT in our work) in image I_i and I_t respectively. N is the number of matched features while ID_i and ID_t are the sequential numbers of I_i and I_t showing the order of sampling. For Eq. 2 to be valid, we assume constant velocity in robot motion. Thresholds are given to the above equations for deciding whether the current view should be considered as a new key frame.

Once a new key frame is detected, we insert its features to the feature pool D and construct a KD-tree over all the features in D. Hence, D always includes the features extracted from all the key frames and the tree is updated every time a new key frame is found. Inspired by the inverted index used in BoW, which keeps the associated image *id*s and word frequencies for each visual word, we

[1]Perceptual aliasing can be intrinsic to the scene or artificially created by the process of image representation, for example, by clustering in creating the visual vocabulary. Our work addresses only the latter source of perceptual aliasing, which can be dealt with using temporal information, i.e. a sequential framework that incorporates information from multiple images.

also maintain a similar table which maps the features in D to their corresponding images from which they are extracted.

B. Indexing Visual Features

Let Q be the set of features extracted from the current image. For each feature in Q, it will go through a tree search in the current KD-tree and the top two nearest neighbors are returned. Distance ratio is applied to determine whether the closest one is a good match or not. If positive, the image containing this matched feature can be immediately retrieved by indexing.

Since KD-tree does not guarantee to find the correct nearest neighbors, the key frame that contains the most matched features is not necessarily the true loop closure key frame. We then perform a second step of further verification of the possible candidates. In this step, top K key frames that contain the most matched features are selected as the candidates and an accurate feature matching in image level is performed between the query image and each candidate. A loop closure is found if the similarity is above some given threshold. This verification is based on the observation that only a few key frames contain matched features after the previous indexing step and the true loop closure key frame is always likely to have multiple matched features by using the tree search.

We also notice that the features in Q that have correct matches in D only represent a small fraction (normally less than 15%) of the total features in Q. This may potentially increase the tolerance of finding the incorrect nearest neighbors. The other important fact is that we are only interested in finding the correct matches for features in Q that *do have* matches in D. In other words, the purpose is to find as many true positive samples as possible using the tree search and the number of false positives is not important. The only possible negative impact of finding a false positive is to retrieve an irrelevant key frame as a candidate after the first step of the algorithm. However, this irrelevant key frame is likely to be discarded since it is unlikely for it to be ranked high among all the key frames with matched features.

C. Loop Closure Detection and Simple SLAM System

The algorithm is summarized in Table I. Here we use $|A|$ to represent the number of elements in set A and variables that are not mentioned in the algorithm have the same meaning as in Eq. (1) and (2). The algorithm is not only limited to loop closure detection but also can be regarded as a simple SLAM system. The similarity measurement is consistent to Eq. (1). In Table I, q is the current image. E is the set of key frames that contain matched features and C is the sorted top K candidates in E according to the number of matched features (if $|E| < K$, we will consider all the key frames). DIST and SIMI are two thresholds used for discarding nearby key frames and determining loop closures.

TABLE I
VISUAL LOOP CLOSURE DETECTION USING KD-TREE

while not the last image **do**
  % step 1: indexing visual features %
  $E =$ matchTree(Q, D);
  for $i = 1 \rightarrow K$ **do**
    if $|ID_q - ID_{E(i)}| \leq$ DIST **then**
      remove $E(i)$ from E;
    end if
  end for
  if $K > |E|$ **then**
    $K = |E|$;
  end if
  $C =$ topK(E);
  % step 2: further verification in image level %
  $i \leftarrow 1$;
  while $i \leq K$ **do**
    $N =$ matchIm(q, $C(i)$);
    $S = \frac{2N}{N_q + N_{C(i)}}$;
    if $S >$ SIMI **then**
      a loop closure is found;
      break;
    end if
    $i \leftarrow i + 1$;
  end while
  % key frame detection if needed %
  if $i > K$ **then**
    isKeyframe = keyframeDetect(q);
    if isKeyframe = TRUE **then**
      $D = D \bigcup Q$;
      updateTree(D);
    end if
  end if
end while

IV. EXPERIMENTAL VALIDATION

In order to validate the performance of the proposed method in visual loop closure detection, several groups of experiments were conducted on a dataset describing an indoor hallway environment. The dataset was sampled by a Pioneer 3-AT mobile robot equiped with a Dragonfly IEEE-1394 camera pointing forward [9]. Images were captured at equal time interval as the robot moved, covering a total distance of approximately 200 meters with two entire loops of the second floor of Computing Science Center, University of Alberta. A total of 7,420 images were collected with resolution of 320×240 pixels. Fig. 1 shows several sample images of the dataset and Fig. 2 is the visualization of the path (unit: meter) according to the odometry readings.

A. Performance of Feature Matching Using KD-tree

Before applying the proposed method to loop closure detection, we would like to first investigate the possibility of using KD-tree in feature matching in terms of both accuracy and run time. We extracted all the SIFT features from the above dataset and randomly selected 500,000 of them as the feature pool D. Another random 20,000 features from the same dataset were used as the test query set Q. By performing a linear search for each query feature over all the features in D, we obtained the ground truth of the test set and 2,415 features out of 20,000 (around 12.1%) had been given corresponding matches. FLANN[2] was used as our implementation of KD-tree [3]. Different numbers of

[2]http://people.cs.ubc.ca/∼mariusm/index.php/FLANN/FLANN

Fig. 1. Sample images of the indoor dataset. The dataset was sampled densely and covers two entire loops of the hallway in the second floor of Computing Science Center, Univeristy of Alberta.

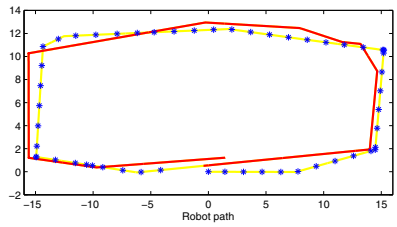

Fig. 2. Visualization of the data describing robot path and key frames according to the odometry readings (after alignment). There are two completed loops. 3,346 images taken in the second loop (red) are considered loop closures. Selected key frames in the first loop (yellow) are shown in blue stars. The starting point is [0, 0]

trees and checked leaf nodes were used in the tree search to illustrate the performance. As discussed previously, we are only curious about how many correct matches we can find for these 2,415 features. Fig. 3 shows that the performance will improve by using multiple trees and checking more leaf nodes in the backtrack process. There is no obvious difference between three and four trees and this suggests that the performance may converge to a number of trees. Hence, it is unnecessary to use too many trees in our method. Notice that the search time will also increase if more trees are used or more leaf nodes are examined. In fact, we used only one tree and compared 10 leaf nodes in our algorithm and found it working pretty well. As shown in the figure, we are still able to get around 90% of matches correct, which is an acceptable result.

Another group of experiments was designed to investigate the applicability of KD-tree in terms of run time. Since the number of key frames is increasing during SLAM process, we are particularly interested in seeing how the speedup factor over linear search evolves with the increase of feature number. Different numbers of features were randomly collected from the same dataset, ranging from 10K to 100K as D. The same test set Q was used as the query samples. There is no overlap between any of the two feature groups

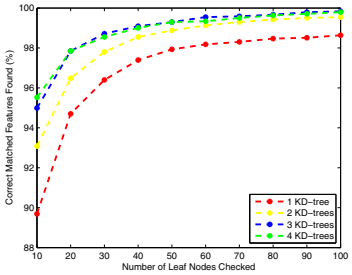

Fig. 3. Result of feature matching with different numbers of trees and checked leaf nodes. The result suggests that in terms of accuracy, KD-tree can be used in loop closure detection. There is no need to use multiple trees since the accuracy is good enough with one tree for our problem.

(including the previous 500,000 feature group). The second row of Table II shows the numbers of matched features between the test set and different pools using exhaustive search as the ground truth. The tree search performance (here one KD-tree was used with 10 leaf nodes checked) in the third row is consistent to the result shown in Fig. 3. Fig. 4 is the speedup factor of KD-tree over linear search on different feature groups. In the tree search, we counted the time of both constructing the tree and performing the search. The result shows that KD-tree is at least three orders of magnitude faster than linear search and the time advantage increases rapidly with the increase number of features.

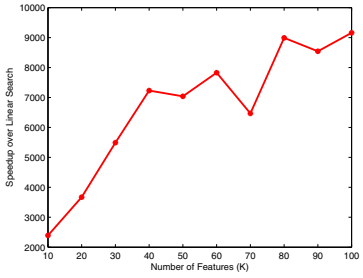

Fig. 4. Speedup factor over linear search vs. different numbers of features. KD-tree is at least three orders of magnitude faster than linear search. It is reasonable to believe that we are able to deal with many key frames in real-time implementation using KD-tree in feature matching.

B. Loop Closure Detection

Based on the above results, we ran the proposed loop closure detection algorithm described in Table I on this indoor dataset on a normal lab computer (2.0 GHz CPU and 4.0 GB memery) and compared the results with BoW of 1,000 visual words in terms of both recall and time (the visual words were generated by clustering more than 20000 features extracted from sampled images of another dataset). The scale dependent feature selection proposed in [9] was also applied here to show the scalability of the method. Selected features were used to build the tree (features in D) and search (features in Q). The run time of the algorithm mainly comes from three aspects: time of building the tree, search time (indexing in step 1) and time

TABLE II

NUMBER OF MATCHED FEATURES AND PERFORMANCE OF TREE SEARCH IN DIFFERENT FEATURE GROUPS

# of features in D	10K	20K	30K	40K	50K	60K	70K	80K	90K	100K
# of matched features (ground truth)	1,747	2,251	2,376	2,529	2,615	2,609	2,819	2,766	2,760	2,688
Correct matched features using KD-tree (%)	88.61	86.94	86.15	87.90	88.83	87.93	86.91	87.20	86.81	87.69

of further verification (step 2). All the other components such as feature extraction and key frame detection are the same to both algorithms and do not need to be considered and compared. Thresholds for Eq. (1) and (2) are 3% and 80 respectively. DIST and SIMI are 100 and 3%, i.e. we do not want to consider the key frames within 100 images in sequential order as the loop closure candidates, and the threshold for accepting a loop closure hypothesis is the same as deciding a key frame. The parameters were all empirically chosen and could be adjustable. The guideline is that the key frames should have a full coverage of the environment but are not redundant in localization. Different groups of parameters were tried and the differences between the proposed method and BoW on different parameter settings were close. A total of 53 key frames were selected (see Fig. 2). We built the ground truth mainly according to the space distances: for each image in the second loop, the closest two key frames are both considered the true loop closure images. This would be subject to a visual check to exclude special cases (e.g. when the robot turns, the two images may look different although they are spatially close). The performance is summarized in Table III for the 3,346 loop closure images in the second loop in terms of recall. As mentioned in Section III, we consider the top K ranked candidates in E to undergo the further verification in step 2. Here we do not worry about the precision because step 2 will reject any candidate that does not have a similarity measurement above the given threshold. In other words, with this step we can consider the precision as 100%. The first column in Table III is the scales above which features were used [9]. So "$>$0" means that the entire original SIFT features were used without applying scale selection. It can be seen that the performance will go down with fewer feature used (more feature selection applied). This is understandable because the number of features is too few. However, our proposed approach can be still superior to BoW even if we reduce the average number of features to only 30 per image (scale$>$4). The top 3 and top 5 results do not differ too much which suggests that the correct loop closure key frames are normally ranked high, with or without feature selection. The run time of both building the tree and indexing is shown in Table IV and V respectively. By using features at scales higher than 4, the average run time is almost one order of magnitude less compared with using the entire set of original features, sacrificing the performance as much as 13%. In most cases, the average time of processing one image is the total time of both search (step 1) and further verification (step 2). The time of rebuilding the KD-tree should be counted in only when a new key frame is found. Therefore our method provides a real-time performance to detect loop closures on this dataset.

TABLE III

RECALL ON INDOOR DATASET WITH SCALE SELECTION (%)

	Top 1	Top 3	Top 5	# of features / image
> 0 (all)	85.12	94.77	95.70	256.6
> 1	82.46	94.17	95.31	211.1
> 2	80.66	91.54	92.17	78.9
> 3	78.81	88.52	88.82	46.4
> 4	72.33	82.85	82.93	30.8
BoW	45.04	72.33	80.39	256.6

To make the results comparable, we also used a single KD-tree with 10 leaf nodes in the backtrack to create the BoW descriptor of an image from the extracted features. Therefore, the time of BoW in Table V is actually the time of converting the image to the BoW descriptor, depending on the vocabulary size and the number of features in the query image. For the top K retrieved candidates with the highest similarity according to their BoW descriptors, the same verification was performed to decide whether a loop closure was found or not. The combined run time of both indexing and verification can be represented as:

$$T = T_i + k \times T_v \tag{3}$$

where T_i is the indexing time shown in Table V and T_v is the time of further verification for each retrieved candidate. k is the number of candidates verified to achieve a certain level of performance. We are only interested in comparing T because all the other parts are the same to both methods as mentioned previously, and the time of verification can be dominant in the total execution time of processing one image, if too many candidates are verified. The verification for each key frame took around 6.9 milliseconds on average. Fig. 5 shows T in both methods with respecct to the performance. It is not surprising to see that BoW spends more time in achieving the same level of performance. In the proposed method, the loop closure key frame is usually included in the top 5 retrieved candidates after the indexing step. However, BoW needs to verify as many as tens of key frames to obtain the correct one. This verification is mainly responsible for the time spent in detecting loop closures.

V. CONCLUSION AND FUTURE WORK

We have presented a loop closure detection method for appearance-only SLAM. Our proposed method does not rely on the widely used BoW representation to extract image descriptors but uses feature matching to improve the recall in detecting loop closures. We used a KD-tree to index visual

TABLE IV

RUN TIME ON INDOOR DATASET - BUILDING THE TREE (MS)

	Min	Max	Average
> 0 (all)	2.1	31.2	16.2
> 1	1.5	26.4	13.6
> 2	0.89	31.2	5.8
> 3	0.66	24.8	3.7
> 4	0.56	26.0	2.8

TABLE V

RUN TIME ON INDOOR DATASET - SEARCH AND INDEXING (MS)

	Min	Max	Average
> 0 (all)	0.55	128.4	10.3
> 1	0.40	97.9	9.0
> 2	0.30	50.8	4.9
> 3	0.27	25.4	3.3
> 4	0.24	16.7	2.3
BoW	0.32	3.7	1.4

features and achieve fast match, followed by a verification step for confirming a true loop closure. It is shown in our work that due to its efficiency in feature matching, KD-tree can be applied in real-time loop closure detection with high recall in an indoor environment. The necessity of using scale dependent feature selection was also discussed for the purpose of extending the work to large outdoor environment. Experimental results from a moderate size indoor dataset confirmed our method in terms of both recall and time.

One limitation of our work is that the online process of building the tree which is tractable on this indoor dataset, but of $O(nlogn)$ complexity with respect to the number of features, will eventually prevent SLAM from real-time implementation in large environment that may contain many thousands of key frames (it took 4 seconds to build a tree over the entire 1.34 million features extracted from 5,000 images in this dataset). In such cases the indexing time in step 1 will also increase. This problem does not exist in BoW since the vocabulary is built offline and there is no time-consuming part in localization and mapping (provided that a low recall is tolerable) regardless how large the environment is. However, there is always a trade-off between the run time and recall

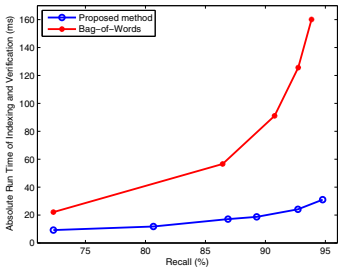

Fig. 5. The absolute run time of indexing and verification vs. recall of both methods. Since BoW requires more key frame candidates to be verified to achieve the same level of performance, it is not as efficient as the proposed method on this indoor dataset in terms of obtaining the same recall.

and this is why we are studying the applicability of feature selection. We leave it as part of our future work to evaluate the proposed method on much larger outdoor environments to see how it can perform.

Another interesting issue is to discover and investigate more efficient feature selection stratergy to keep the performance as high as possible while reducing the feature number. Inserting the features in the current tree instead of rebuilding it every time is also an alternative to reduce the time. In addition, the current framework of implementing SLAM is somewhat naïve. Actually it deals with localization only. We are planning to improve the system by taking into account the update of the map when a loop closure is found to make it fully functional in both localization and mapping.

REFERENCES

[1] J. Sivic and A. Zisserman, "Video Google: A Text Retrieval Approach to Object Matching in Videos", *IEEE International Conference on Computer Vision*, Nice, France, pp. 1470-1477, 2003.

[2] J. Wang, J. Yang, K. Yu, *et al.*, "Locality-constrained Linear Coding for Image Classification", *IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, USA, pp. 3360-3367, 2010.

[3] M. Muja and D. G. Lowe, "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration", *International Conference on Computer Vision Theory and Applications*, Lisboa, Portugal, 2009.

[4] P. Newman, D. Cole and K. Ho, "Outdoor SLAM using Visual Appearance and Laser Ranging", *IEEE International Conference on Robotics and Automation*, Orlando, USA, pp. 1180-1187, 2006.

[5] M. Cummins and P. Newman, "FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance", *The International Journal of Robotics Research*, 27(6), pp. 647-665, June 2008.

[6] M. Cummins and P. Newman, "Highly Scalable Appearance-Only SLAM FAB-MAP 2.0", *Robotics: Science and Systems*, Seattle, USA, 2009.

[7] A. Kawewong, N. Tongprasit, S. Tangruamsub, *et al.*, "Online and Incremental Appearance-based SLAM in Highly Dynamic Environments", *The International Journal of Robotics Research*, Published online June 2010.

[8] H. Zhang, B. Li and D. Yang, "Keyframe Detection for Appearance-Based Visual SLAM", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, pp. 2071-2076, 2010.

[9] H. Zhang, "Loop-Closure Detection with Scale Invariant Visual Features", *IEEE International Conference on Robotics and Automation*, Shanghai, China, pp. 3125-3130, 2011.

[10] X. Wang and H. Zhang, "Good Image Features for Bearing-only SLAM", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, Chinae, pp. 2576-2581.

[11] A. Angeli, D. Filliat, S. Doncieux, *et al*, "Real-Time Visual Loop-Closure Detection", *IEEE International Conference on Robotics and Automation*, Pasadena, USA, pp. 1842-1847, 2008.

[12] A. Angeli, D. Filliat, S. Doncieux, *et al.*, "Fast and Incremental Method for Loop-Closure Detection Using Bags of Visual Words", *IEEE Transactions on Robotics*, 24(5), pp. 1027-1037, 2008.

[13] A. Angeli, D. Filliat, S. Doncieux, *et al.*, "Incremental vision-based topological SLAM", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nice, France, pp. 1031-1036, 2008.

[14] A. Angeli, D. Filliat, S. Doncieux, *et al*, "Visual topological SLAM and global localization", *IEEE International Conference on Robotics and Automation*, Kobe, Japan, pp. 4300-4305, 2009.

[15] D. Nistér and H. Stewénius, "Scalable Recognition with a Vocabulary Tree", *IEEE Conference on Computer Vision and Pattern Recognition*, New York, USA, pp. 2161-2168 , 2006.

[16] J. Philbin, O. Chum, M. Isard, *et al.*, "Object retrieval with large vocabularies and fast spatial matching", *IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, USA, 2007.

[17] C. Silpa-Anan and R. Hartley, "Optimised KD-trees for fast image descriptor matching", *IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, USA, 2008.

[18] D. G. Lowe, "Distinctive image features from scale-invariant keypoints", *International Journal of Computer Vision*, 60(2), pp. 91-110, 2004.