Keyframe-Based Real-Time Camera Tracking

Zilong Dong[1] Guofeng Zhang[1*] Jiaya Jia[2] Hujun Bao[1]

[1]State Key Lab of CAD&CG, Zhejiang University
{zldong, zhangguofeng, bao}@cad.zju.edu.cn

[2]The Chinese University of Hong Kong
leojia@cse.cuhk.edu.hk

Abstract

We present a novel keyframe selection and recognition method for robust markerless real-time camera tracking. Our system contains an offline module to select features from a group of reference images and an online module to match them to the input live video in order to quickly estimate the camera pose. The main contribution lies in constructing an optimal set of keyframes from the input reference images, which are required to approximately cover the entire space and at the same time minimize the content redundancy amongst the selected frames. This strategy not only greatly saves the computation, but also helps significantly reduce the number of repeated features so as to improve the camera tracking quality. Our system also employs a parallel-computing scheme with multi-CPU hardware architecture. Experimental results show that our method dramatically enhances the computation efficiency and eliminates the jittering artifacts.

1. Introduction

Vision-based camera tracking aims to estimate camera parameters, such as rotation and translation, based on the input images (or videos). It is a foundation for solving a wide spectrum of computer vision problems, e.g., 3D reconstruction, video registration and enhancement. Offline camera tracking for uncalibrated image sequences can achieve accurate camera pose estimation [14, 23, 29] without requiring high efficiency. Recently, real-time markless tracking [9, 28, 19, 18] has attracted much attention, as it finds many new applications in augmented reality, mobility, and robotics.

This paper focuses on developing a practical realtime camera tracking system using the global localization (GL) scheme [28, 26], which involves an offline process for space abstraction using features and an online step for feature matching. Specially, the offline step extracts sparse invariant features from the captured reference images and uses them to represent the scene. The 3D locations of these invariant features can be estimated by offline structure-from-motion (SFM). Afterwards, taking these features as references, the successive online process is to match them with the features extracted from the captured live video for establishing correspondences and quickly estimating new camera poses.

GL scheme is robust to fast movement because it matches features in a global way. This also precludes the possibility of error accumulation. It, however, has the following common problems in prior work. First, it is difficult to achieve real-time performance due to expensive feature extraction and matching, even in a relatively small working space. Second, these methods rely excessively on the feature distinctiveness, which cannot be guaranteed when the space scale is getting large or the scene contains repeated structures. It was observed that the matching reliability descends quickly when the number of features increases, which greatly affects the robustness and practicability of this system in camera tracking.

In this paper, we solve the above efficiency and reliability problems and develop a complete real-time tracking system using the GL scheme. Our contribution lies in the following three ways. First, we propose an effective keyframe-based tracking method to increase its practicability in general camera tracking for large scale scenes. A novel keyframe selection algorithm is proposed to effectively reduce the online matching ambiguity and redundancy. These keyframes are selected from all reference images to abstract the space with a few criteria: i) the keyframes should be able to approximate the original reference images and contain as many salient features as possible; ii) the common features among these frames are minimum in order to reduce the feature non-distinctiveness in matching; iii) the features should be distributed evenly in the keyframes such that given any new input frame in the same environment, the system can always find sufficient feature correspondences and compute the accurate camera parameters.

Second, with the extracted keyframes, in the real-time camera tracking stage, we contribute an extremely efficient keyframe recognition algorithm, which is able to find ap-

*Correspondence author: Guofeng Zhang

propriate matching keyframes almost instantly from hundreds of candidates. Therefore, the computation can be greatly saved compared to the conventional global feature matching. Finally, we develop a parallel-computing framework for further speed-up. Realtime performance is yielded with all these contributions in our key-frame camera tracking system.

2. Related Work

We review camera tracking methods using keyframes and feature-based location recognition in this section.

Real-time Camera Tracking In the past a few years, SLAM was extensively studied [11, 4, 10, 16, 17] and used for real-time camera tracking. SLAM methods estimate the environment structure and the camera trajectory online, under a highly nonlinear partial observation model. They typically use frame-to-frame matching and confining-search-region strategies for rapid feature matching. As a result, they usually run fast. However, drifting and relocalisation problems could be produced with this scheme because it highly relies on the accuracy of past frames. Recent development mainly concentrated on improving the robustness and accuracy in a larger scale scene. The major issues of this scheme include relocalisation [32, 3, 17, 12] after camera lost, submap merging and switching [2], and the close loop management [31].

If 3D representation for the space is available, real-time camera tracking can be easier and more robust. Several markerless algorithms [30, 6] have been proposed to employ the object's CAD model to facilitate camera pose estimation. However, these CAD models are usually difficult, if not impossible, to be constructed. Skrypnyk and Lowe [28] proposed modeling natural scene using a set of sparse invariant features. The developed system contains two modules, i.e. the offline feature-based scene modeling and online camera tracking. It runs at low frame rate due to expensive SIFT feature extraction and matching. It also highly relies on the distinctiveness of SIFT features, and is therefore limited to representing a relatively small space. This paper focuses on solving these problems in a two-stage tracking system.

Keyframe-based Methods Keyframe selection is a common technique to reduce the data redundancy. In the real-time camera tracking method of Klein et al. [16], a set of online keyframes were selected, which facilitate the bundle adjustment for the 3D map recovery. In [17], the keyframes were used for relocalisation with simple image descriptors. For model-based tracking, Vacchetti et al. [30] selected keyframes manually in the offline mode, and match each online frame to a keyframe with the closest visible area. In all these methods, keyframes are selected manually or by a simple procedure, which is not optimal for the tracking task when the camera undergoes complex motions. In our method, a set of optimal keyframes are selected for representing and abstracting a space. They are vital for efficient online feature matching.

Feature-based Location Recognition There are approaches to employ the invariant features for object and location recognition [27, 24, 25, 7, 8, 1]. These methods typically extract invariant features for each image, and use them to estimate the similarity of different images. For effectively dealing with a large-scale image databases, a vocabulary tree [22, 5] was adopted to organize and search millions of feature descriptors. However, these methods do not extract sparse keyframes to reduce the data redundancy, and cannot be directly used for real-time tracking. The appearance-based SLAM method of Cummins and Newman [8] considers the inter-dependency among the features and applies the bag-of-words method within a probabilistic framework to increase the speed of location recognition. However, the computation cost is still high and it cannot achieve real-time tracking.

Recently, Irschara *et al*. [15] propose a fast location recognition technique based on SFM point clouds. In order to reduce the 3D database size to improve recognition performance, synthetic views are involved to introduce a compressed 3D scene representation. In contrast, we propose constructing an optimal set of keyframes from the input reference images by minimizing an energy function, which establishes a good balance between the representation completeness and the redundancy reduction.

3. Framework Overview

We first give an overview of our framework in Table 1. It contains two modules, i.e. the offline feature-based scene modeling and online camera tracking. The offline module is responsible for processing the *reference images* and modeling space with sparse 3D points. In this stage, SIFT features [21] are first detected from the reference images to establish the multi-view correspondences. Then we use the SFM method [33] to estimate the camera pose together with the 3D locations of the SIFT features.

1. **Offline space abstraction**:
1.1 Extract SIFT features from the reference images, and recover their 3D positions by SFM.
1.2 Select optimal keyframes and construct a vocabulary tree for online keyframe recognition.
2. **Online real-time camera tracking**:
2.1 Extract SIFT features for each input frame of the incoming live video.
2.2 Quickly select candidate keyframes.
2.3 Feature matching with candidate keyframes.
2.4 Estimate camera pose with the matched features.

Table 1. *Framework Overview*

In the online module, we estimate the camera parameters for each input frame in real-time given any captured live video in the same space. Instead of frame-by-frame matching using all reference images, in our approach, we select several optimal keyframes to represent the scene, and build a vocabulary tree for online keyframe recognition. For each online frame, we select appropriate candidate keyframes by a fast keyframe recognition algorithm. Then the live frame only needs to be compared with the candidate keyframes for feature matching. With the estimated 3D positions of all reference features on keyframes and sufficient 2D-3D correspondences for the live frame, the camera pose can be reliably estimated.

4. Optimal Keyframe Selection

As described above, directly using all reference images for feature detection and online matching is not optimal. To handle images taken with the camera moving in a moderate-scale scene, we propose selecting keyframes in order to increase the feature detection efficiency and reduce the possible matching ambiguity. Our thought is to select an optimal subset of reference frames to approximate the 3D space. These frames should contain as many salient features as possible and at the same time make the features uniformly distributed in the environment.

So we define the problem as follows. Given input n reference images $\hat{I} = \{I_i | i = 1, 2, ..., n\}$, we attempt to compute an optimal subset (i.e. keyframe set) $F = \{I_k | k = i_1, i_2, ...i_K\}$, which minimizes the cost defined in the function $E(F; \hat{I})$. The keyframe number K is adaptive in our method to maximize the flexibility. $E(F; \hat{I})$ consists of two terms, i.e., the completeness term $E_c(F)$ and the redundancy term $E_r(F)$, modeling respectively the scene completeness and pixel redundancy:

$$E(F; \hat{I}) = E_c(F) + \lambda E_r(F), \tag{1}$$

where λ is a weight. The definitions of the two terms are described as follows.

4.1. Completeness Term

The completeness term is used to constrain that the selected keyframes contain as many salient SIFT features as possible. For real-time tracking, we require that one such feature can find multiple matches in different frames so as to accurately compute its 3D coordinates.

All matched SIFT features are with the similar descriptors [21]. For reasons of efficiency, we cluster them and unify their representation by averaging the SIFT descriptors (we use the 64D descriptors). Each of the resulted feature clusters is denoted as $\mathcal{X}$, which contains a series of matched features in multiple frames, written as $\mathcal{X} = \{\mathbf{x}_i | i \in f(\mathcal{X})\}$ where $f(\mathcal{X})$ denotes the reference image set spanned by $\mathcal{X}$. It is notable that $|f(\mathcal{X})|$, for all $\mathcal{X}$, must be larger than 1 because if one feature finds no match in other images, its 3D position cannot be determined. We denote $f(\mathcal{X})$ where $|f(\mathcal{X}) \geq l|$ as *superior features* and the set of the superior features as $V(\hat{I})$. l is set to $10 \sim 20$ in our experiments.

We define the saliency of one SIFT feature as the combination of two factors, i.e. the match count of one feature in different reference images $|f(\mathcal{X})|$ and the Difference-of-Gaussian (DoG) strength, and write it as

$$s(\mathcal{X}) = D(\mathcal{X}) \cdot \min(|f(\mathcal{X})|, T), \tag{2}$$

where T is the truncated threshold to prevent a long track over-suppress others. It is set to 30 in our experiments. A large value in $|f(\mathcal{X})|$ indicates a high confidence of matching. $D(\mathcal{X})$ is expressed as

$$D(\mathcal{X}) = \frac{1}{|f(\mathcal{X})|} \sum_{i \in f(\mathcal{X})} D_i(\mathbf{x}_i),$$

where D_i denotes the DoG map [21]. $D(\mathcal{X})$ represents the average DoG map for all features in $f(\mathcal{X})$. The larger $D(\mathcal{X})$ is, the higher saliency the feature set $\mathcal{X}$ has.

Despite the above two measures, another important constraint to make real-time tracking reliable is the spatially near-uniform distribution of all features in the space. It is essential for finding sufficient matches with respect to input online frames in the same space.

We define the feature density $d(\mathbf{y}_i)$ for each pixel $\mathbf{y}$ in image i. Its computation is described in Algorithm 1. With the density maps for all images, we define the *set density* as

$$d(\mathcal{X}) = \frac{1}{|f(\mathcal{X})|} \sum_{i \in f(\mathcal{X})} d(\mathbf{x}_i),$$

where $d(\mathbf{x}_i)$ denotes the feature density of $\mathbf{x}_i$ in image i.

Algorithm 1 Feature density computation for image i

1. Initialize all densities to zeros in the map.
2. for $j = 1, ..., m$, $\%$ m is the feature number in image i
   for each pixel $\mathbf{y}_i \in W(\mathbf{x}_j)$,
   $\%W$ is a 31×31 window and
   $\%\mathbf{x}_j$ is the coordinate of feature j in image i
   $d(\mathbf{y}_i)$ += 1.

Finally, our completeness term is defined as:

$$E_t(F) = 1 - \Big(\sum_{\mathcal{X} \in V(F)} \frac{s(\mathcal{X})}{\eta + d(\mathcal{X})}\Big) / \Big(\sum_{\mathcal{X} \in V(\hat{I})} \frac{s(\mathcal{X})}{\eta + d(\mathcal{X})}\Big), \tag{3}$$

where η controls the sensitivity to feature density. It is set to 3 in our experiments. $V(F)$ denotes the superior feature set in the keyframe set F.

4.2. Redundancy Term

The common features in different keyframes should be as small as possible in order to reduce the redundancy of features and simplify the extensive feature matching. Since we have detected feature set $f(\mathcal{X})$, the redundancy minimization problem is equivalent to making the features in the same $f(\mathcal{X})$ distributed to a minimum set of keyframes. We therefore define

$$E_r(F) = \frac{1}{|V(\hat{I})|} \sum_{\mathcal{X} \in V(F)} (|f(\mathcal{X}) \cap F| - 1), \qquad (4)$$

where $1/|V(\hat{I})|$ is for normalization, $|f(\mathcal{X}) \cap F|$ computes the copies of features in both $\mathcal{X}$ and the keyframes. $|f(\mathcal{X}) \cap F| = 1$ indicates no redundancy.

4.3. Keyframe Selection

With an exhaustive search of all possible subsets of $\hat{I}$ in the reference images, we can certainly find the optimal set of keyframes that minimizes the cost in (1). However, it is not computationally efficient to enumerate the 2^n subsets. In [20], with a fixed number of keyframes, dynamic programming (DP) was used to search the optimal solution for video summarization. Note that this scheme does not suit our system because our cost function has a much more complex form and the number of keyframes is not fixed. Further, the method in [20] assumes that only adjacent frames are possibly overlapped, and accordingly proposed a greedy algorithm to approximate the solution. We do not make the same assumption in problem definition.

Our keyframe selection process is based on a steepest-descent-like method as described in Algorithm 2. It proceeds in the following way. To begin with, we construct an empty F and then progressively add frames. In each pass, a new keyframe that reduces the *most* energy is added to F. This process continues until the cost cannot be reduced anymore. The computation complexity is $O(n^2)$. In our experiments, it takes only a few seconds to find keyframes from hundreds or thousands of images, and the obtained keyframes are always sufficiently good for the purpose of real-time camera tracking.

Algorithm 2 Keyframe selection

1. Let $F = \emptyset$.
2. If $\forall I_i \in \{\hat{I} \setminus F\}, E(F \cup \{I_i\}) \geq E(F)$, exit.
3. Otherwise, $I' = \arg\min_{I_i \in \{\hat{I} \setminus F\}} E(F \cup \{I_i\})$, and $F = F \cup \{I'\}$, go to step 2.

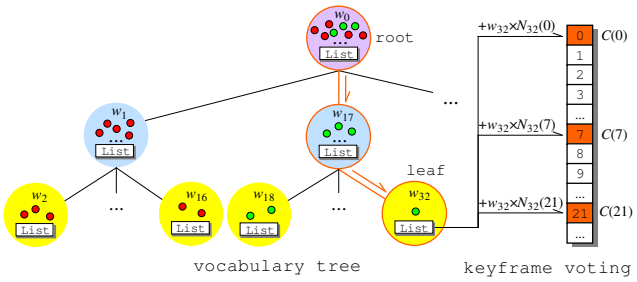


Figure 1. *A vocabulary tree. All feature descriptors are originally in the root node, and are partitioned hierarchically. Each node has a weight to represent its distinctiveness.*

5. Fast Keyframe Recognition

With the collected keyframes, we perform keyframe-based feature matching for online tracking. However, it is still costly to find all matches between the input frame and the keyframes, especially when there exist a considerable amount of keyframes. We observe that any input target frame only covers a small portion of the space; so it is inevitable that many keyframes do not even have the common content with the input. We thus exclude these useless frames from feature matching to save computation, by employing a vocabulary based fast selection algorithm.

5.1. Vocabulary Tree Construction

Given a set of keyframes, we construct a visual vocabulary by hierarchically clustering all the descriptors of superior features. Our vocabulary tree construction is similar to that of [22, 25] where the vocabulary V is organized as an l level tree with branching factor b, and the root node (cluster) contains all descriptors. The K-Means method is used to partition the descriptors into b clusters; then all clusters become children of the root node. This process continues, recursively partitioning children nodes until a specified level l is reached. The final vocabulary tree has $|V|$ nodes, and each node i is attached with a mean descriptor of the features in the node. Figure 1 gives an illustration. Each node i has a keyframe list L_i. $N_i(k)$ denotes the number of features in keyframe k that are clustered in node i.

Each node i also contains a weight w_i, which represents its distinctiveness. In our system, the weight is defined as

$$w_i = \log \frac{K}{|L_i|}, \qquad (5)$$

where K is the number of all keyframes. The node count $|V|$ is determined by the branching factor b and tree depth l. In our experiments, we normally select $20 \sim 50$ keyframes, and each keyframe extracts about $300 \sim 500$ features. Therefore, we usually set $b = 8, l = 5$ in our experiments.

Algorithm 3 Candidate keyframe selection

1. Set the matching value $C(k) = 0$ for each keyframe k.
2. For each online frame, the detected m features are matched from the root node to leafs in the vocabulary tree as follows:

   In each level, for each closest node i with weight $w_i > \tau$,

   for each $k \in L_i$,

   $C(k) += N_i(k) \cdot w_i$.
3. Select $\mathcal{K}$ keyframes with largest C.

5.2. Candidate Keyframe Searching and Matching

In [22, 25], an appearance vector is used to describe an image, where each element corresponds to one node in the vocabulary tree. To construct the appearance vector of an image, each feature descriptor is simply searched down the tree by at each level comparing the descriptor vector to the b candidate cluster centers and choosing the the closest one. Then the weight of the closest node in each level is added to the corresponding elements of the appearance vector. This process repeats until all m features in the image are processed.

With the built appearance vectors, the similarity of two images can be estimated with vector distance computation. Directly employing this strategy for keyframe recognition, even with the sparse property of the appearance vectors (i.e. only compare non-zero elements), results in computation time $O(m \cdot K)$, which grows linearly with the number of keyframes.

Here, we introduce a more efficient keyframe recognition algorithm (Algorithm 3 with an illustration in Figure 1). We employ a voting scheme similar to the one of [1]. The computational complexity is $O(m \cdot \tilde{L})$, where $\tilde{L}$ is the average keyframe number of the matched nodes in traversing. Specifically, in Algorithm 3, we define a threshold τ to exclude the non-distinctive nodes which are shared by many keyframes. In experiments, we observe that the closer a node is to the root, the easier it is excluded because these top-level nodes most likely contain redundant features. On the contrary, $\tilde{L}$ is usually a very small value for leaf nodes. This weighting scheme makes the time spent on keyframe recognition almost constant even with a large number of keyframes. The majority of computation is on descriptor comparison. It is stable because the feature number m in each online frame, branching factor b, and tree depth l seldom change drastically.

After selecting the most related key frames for an online image, we perform feature matching. This is quick because an offline KD-tree is independently constructed for each keyframe, which can speed up matching. The outliers are rejected in our system by epipolar geometry between the online frame and keyframes using RANSAC [13]. To obtain more matches, we can utilize matched features on the previous frame by finding their correspondences on the current frame through local spatial searching [16]. Once all matches are found, since a feature in the reference keyframes corresponds to a 3D point in the space, we use these 2D-3D correspondences to estimate camera pose [28].

Module	Time per frame
SIFT feature extraction	$\approx$ 110 ms
Keyframe Recognition	$\approx$ 6 ms
Keyframe-based matching	$\approx$ 15 ms
Camera pose estimation	$\approx$ 20 ms
Rendering	$\approx$ 10 ms

Table 2. *Process time per frame with a single thread.*

6. Implementation and Results

In this section, we describe the implementation details and show our experimental results.

6.1. Parallel Computing

Frame rate and latency are two key indexes in a real-time vision system. Frame rate is usually written as the number of frames that can be processed in one second, and latency is the elapsed time between capturing and rendering a frame. A good real-time system should have high frame rate and low latency. Table 2 shows the time spent in different steps for one input frame of the 'cubicle' example. Even though we only use a single working thread, the latency is as small as $160ms$, and the frame rate is around 6 fps. This validates the efficiency of our keyframe-based tracking scheme.

We also employ a parallel computing technique using a multi-core CPU to improve the system performance. Our framework contains two parallel hierarchies – that is, the inter-frame and intra-frame computations. For inter-frame process, we assign the computation tasks for each frame to separate threads. Therefore, the frame rate could be several times higher than using a single CPU. However, the latency is not reduced because the total computation time for each frame does not decrease. To tackle it, we assign the most expensive computation, i.e. feature extraction and matching, to multiple threads. Since the features are independent of each other, their description can be generated simultaneously on multiple threads, and once one of them is ready, it can be sent to matching immediately. With this intra-frame parallelism, the latency caused by feature extraction is significantly reduced, and the latency caused by matching can basically be ignored. Figure 2 illustrates the parallel computing diagram. All the modules are connected and syn-

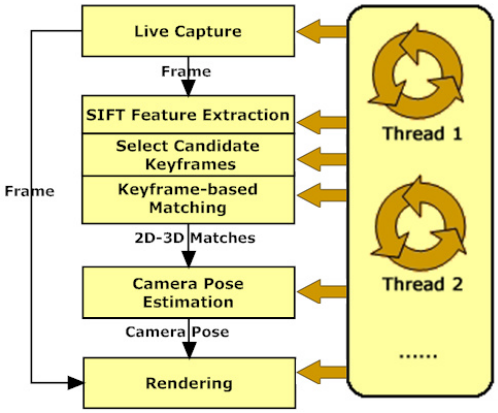


Figure 2. *Diagram of our system. It is divided to multiple parts connected by thread-safe buffers. Different components run on separate working threads, and are synchronized by the frame time stamp.*

chronized by thread-safe buffers. Our system can operate at about 20 fps.

6.2. Experimental Results

All results shown in this section and the supplementary video [1] are produced on a computer with a 4-core Xeon 2.66GHz CPU. The reference and live frames (resolution: 640×480) are captured by a Logitech Quick-Cam Pro 9000 video camera.

We first show an indoor cubicle example in Figure 3. Figure 3(a) shows the recovered 6167 sparse 3D points in the offline stage. Figure 3(c) shows the selected keyframes by our method. There are only a few common features among these frames and they basically cover the entire space, as shown in Figure 3(b). The original reference frames, the captured live frames, and the real-time rendering of the system are illustrated in the supplementary video.

Table 3 shows how setting different λ would influence keyframe selection. It can be observed that if we select 33 keyframes, more than 99% matched features in all reference images are covered in this keyframe set, and almost no loss can be caused when we use them to estimate the camera poses for online frames. Even with only 13 keyframes, about 70% features in the original reference images can be maintained. The keyframe sparsity makes online feature matching robust and fast. In our experiments, we set $\lambda = 0.1$, since it maintains most features.

Our keyframe recognition is very efficient, which spends only $6ms$ even with a single working thread. Figure 4 shows the performance comparison. Compared to the appearance-vector-based method [22], our running time is less variant to the change of the keyframe numbers.

[1]The supplementary video as well as the complete sequences can be downloaded from the following website: **http://www.cad.zju.edu.cn/home/gfzhang/projects/realtimetracking/**

λ	**Keyframe Number**	**Feature Completeness**	**Feature Redundancy** E_r
0.01	33	99.64%	0.748824
0.05	28	97.36%	0.503486
0.1	23	93.06%	0.319604
1	13	69.09%	0.063239
10	8	48.58%	0.000649

Table 3. *The statistics of feature completeness and redundancy with different λ and keyframe numbers.*

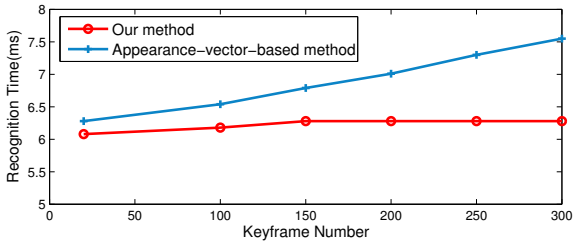


Figure 4. *Time spent in keyframe recognition. The computation of the appearance-vector-based method [22] grows rapidly with the keyframe number while our method does not. The total running time of our method is short.*

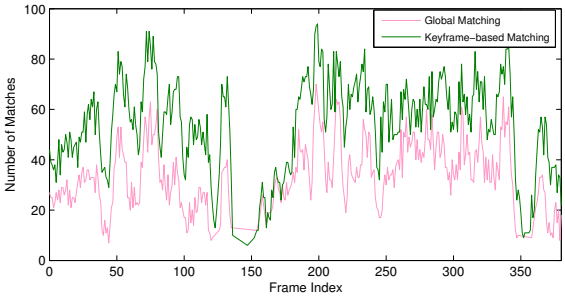
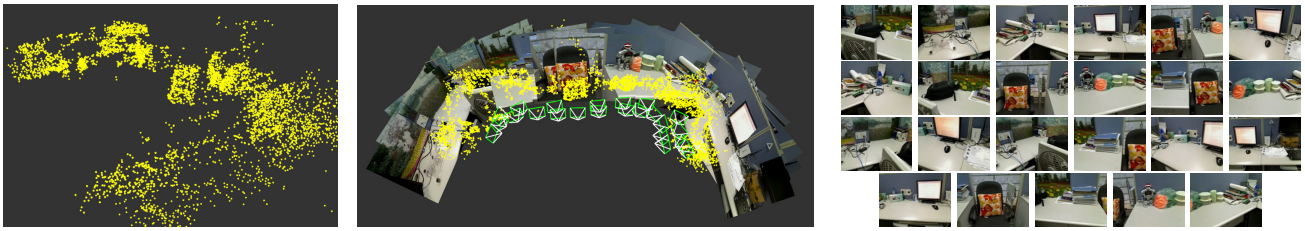


Figure 5. *Comparison of feature matching. Our keyframe-based method yields much more reliable matches than global matching.*

For demonstrating the effectiveness of our method, we also compare our keyframe-based matching with the method of [28]. In [28], a KD tree was constructed for all reference features. Each feature in a live frame is compared with those in the KD tree. We name this scheme *global matching* to distinguish it from our keyframe-based matching. Figure 5 compares the real-time matching quality between these two methods in the indoor cubicle example. It is measured by the number of correct matches in processing one online frame. As illustrated, our keyframe method yields much more reliable matches than the global method. With the KD-tree, the matching time of each feature is $O(\log M)$ for global matching, where M is the total number of features. For our keyframe-based matching, the running time is only $\mathcal{K} \cdot O(\log m)$, where m is the average feature number in each keyframe and $\mathcal{K}$ is the number of the candidate keyframes.

For global matching, the computation time grows with



(a) The recovered 3D points in the offline stage. (b) The selected keyframes viewed in 3D. (c) The keyframes.

Figure 3. *The indoor cubicle example.*

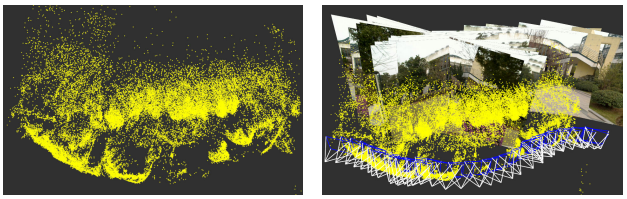
M. But for our keyframe-based method, its computation time is relatively more stable and does not grow with the total number of features. In our experiments, for each online frame, we extract about 300 SIFT features, and the global matching time is about $40ms$ ($M = 6167$) with a single working thread. Our method only uses $15ms$ with $\mathcal{K} = 4$. The augmented result by inserting a virtual object is shown in our supplementary video. The jittering artifact when inserting an object into the live video is noticeable with global matching, but not with our keyframe-based matching.

Figure 6 shows another example of an outdoor scene, containing many repeated similar structures. The scale of the space is relatively large. Figure 6(a) shows the recovered 61522 3D points. The selected 39 keyframes are shown in Figure 6(b), covering 53.7% of the superior features. Please refer to our supplementary video for augmented result and more examples.

Klein and Murray [16, 17] employed online bundle adjustment (BA) with parallel computing to avoid offline 3D reconstruction. This strategy, however, is not very suitable for our examples that are taken in relatively large scenes because SFM for such a workspace requires computationally-expensive global BA which cannot be done online. We have tested our sequences using the publicly accessible code PTAM (http://www.robots.ox.ac.uk/∼gk/PTAM/). The input frame rate is set to 5fps to give enough time to the local BA thread, and each sequence is repeated for the global BA to converge. Even so, we found PTAM only succeeded in tracking the first half part of the indoor cubicle sequence, and failed on the other two outdoor sequences. Readers are referred to our supplementary video for the detailed comparison.

7. Conclusions

In this paper, we have presented an effective keyframe-based real-time camera tracking. In the offline stage, the keyframes are selected from the captured reference images based on a few criteria. For quick online matching, we introduce a fast keyframe candidate searching algorithm to avoid exhaustive frame-by-frame matching. Our experiments show that a small number of candidate reference images are sufficient for achieving high coverage of features in the input images. Compared to global matching, our method not only simplifies feature matching and speeds it up, but also minimizes the matching ambiguity when the original images contain many non-distinctive features. It makes camera pose estimation robust.



(a) The recovered 3D feature points. (b) The keyframes viewed in 3D.

Figure 6. *An outdoor scene example.*

Our method still has limitations. If the camera moves to a place significantly different from the training keyframes, the camera pose cannot be accurately estimated. In practice, this problem can be alleviated by capturing sufficient reference images to cover the space. In addition, this paper has demonstrated the effectiveness of the proposed keyframe selection and recognition methods. We believe it could be combined with other frame-to-frame tracking scheme, like SLAM, to further improve the efficiency. So one possible direction of our future work is to reduce offline processing by collecting online keyframes to update and extend space structure, and combine frame-to-frame tracking (such as the SLAM methods) to further improve the performance of our system in an even larger space. Employing GPU for further acceleration will be explored.

Acknowledgements

This work is supported by the 973 program of China (No. 2009CB320804), NSF of China (No. 60633070), the 863 program of China (No. 2007AA01Z326), and the Research Grants Council of the Hong Kong Special Administrative Region, under General Research Fund (Project No. 412708).

References

[1] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer. A fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions on Robotics, Special Issue on Visual Slam*, October 2008. 2, 5

[2] R. O. Castle, G. Klein, and D. W. Murray. Video-rate localization in multiple maps for wearable augmented reality. In *Proc 12th IEEE Int Symp on Wearable Computers, Pittsburgh PA*, 2008. 2

[3] D. Chekhlov, W. Mayol-Cuevas, and A. Calway. Appearance based indexing for relocalisation in real-time visual slam. In *19th Bristish Machine Vision Conference*, pages 363–372. BMVA, September 2008. 2

[4] D. Chekhlov, M. Pupilli, W. Mayol, and A. Calway. Robust Real-Time Visual SLAM Using Scale Prediction and Exemplar Based Feature Description. In *CVPR*, pages 1–7, 2007. 2

[5] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *ICCV*, pages 1–8, 2007. 2

[6] A. I. Comport, E. Marchand, M. Pressigout, and F. Chaumette. Real-time markerless tracking for augmented reality: The virtual visual servoing framework. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):615–628, July-August 2006. 2

[7] M. Cummins and P. Newman. Fab-map: Probabilistic localization and mapping in the space of appearance. *Int. J. Rob. Res.*, 27(6):647–665, 2008. 2

[8] M. J. Cummins and P. Newman. Accelerated appearance-only slam. In *ICRA*, pages 1828–1833, 2008. 2

[9] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *ICCV*, pages 1403–1410, 2003. 1

[10] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):1052–1067, 2007. 2

[11] E. Eade and T. Drummond. Scalable monocular slam. In *CVPR (1)*, pages 469–476, 2006. 2

[12] E. Eade and T. Drummond. Unified loop closing and recovery for real time monocular slam. In *British Machine Vision Conference (BMVC)*, 2008. 2

[13] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981. 5

[14] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004. 1

[15] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. In *CVPR*, 2009. 2

[16] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *ISMAR 2007*, pages 225–234, Nov. 2007. 2, 5, 7

[17] G. Klein and D. Murray. Improving the agility of keyframe-based slam. In *ECCV*, volume 2, pages 802–815, 2008. 2, 7

[18] T. Lee and T. Höllerer. Hybrid feature tracking and user interaction for markerless augmented reality. In *VR*, pages 145–152, 2008. 1

[19] V. Lepetit and P. Fua. Monocular model-based 3D tracking of rigid objects. *Found. Trends. Comput. Graph. Vis.*, 1(1):1–89, 2005. 1

[20] T. Liu and J. R. Kender. Optimization algorithms for the selection of key frame sequences of variable length. In *ECCV (4)*, pages 403–417, 2002. 4

[21] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. 2, 3

[22] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, pages 2161–2168, Washington, DC, USA, 2006. IEEE Computer Society. 2, 4, 5, 6

[23] M. Pollefeys, L. V. Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):207–232, 2004. 1

[24] F. Schaffalitzky and A. Zisserman. Automated location matching in movies. *Computer Vision and Image Understanding*, 92(2-3):236–264, 2003. 2

[25] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *CVPR*, 2007. 2, 4, 5

[26] S. Se, D. Lowe, and J. Little. Vision-based global localization and mapping for mobile robots. *IEEE Transactions on Robotics*, 21:364– 375, 2005. 1

[27] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, page 1470, Washington, DC, USA, 2003. IEEE Computer Society. 2

[28] I. Skrypnyk and D. G. Lowe. Scene modelling, recognition and tracking with invariant image features. In *ISMAR '04: Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 110–119, Washington, DC, USA, 2004. IEEE Computer Society. 1, 2, 5, 6

[29] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from Internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, November 2008. 1

[30] L. Vacchetti, V. Lepetit, and P. Fua. Combining edge and texture information for real-time accurate 3D camera tracking. In *Third IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 48–57, Arlington, Virginia, Nov 2004. 2

[31] B. Williams, J. Cummins, M. Neira, P. Newman, I. Reid, and J. Tardos. An image-to-map loop closing method for monocular SLAM. In *Proc. International Conference on Intelligent Robots and and Systems*, 2008. 2

[32] B. Williams, G. Klein, and I. Reid. Real-Time SLAM Relocalisation. In *ICCV*, pages 1–8, 2007. 2

[33] G. Zhang, X. Qin, W. Hua, T.-T. Wong, P.-A. Heng, and H. Bao. Robust metric reconstruction from challenging video sequences. In *CVPR*, 2007. 2