Robust Monocular SLAM in Dynamic Environments

Wei Tan Haomin Liu Zilong Dong Guofeng Zhang* Hujun Bao*
State Key Lab of CAD&CG, Zhejiang University

Abstract

We present a novel real-time monocular SLAM system which can robustly work in dynamic environments. Different to the traditional methods, our system allows parts of the scene to be dynamic or the whole scene to gradually change. The key contribution is that we propose a novel online keyframe representation and updating method to adaptively model the dynamic environments, where the appearance or structure changes can be effectively detected and handled. We reliably detect the changed features by projecting them from the keyframes to current frame for appearance and structure comparison. The appearance change due to occlusions also can be reliably detected and handled. The keyframes with large changed areas will be replaced by newly selected frames. In addition, we propose a novel prior-based adaptive RANSAC algorithm (PARSAC) to efficiently remove outliers even when the inlier ratio is rather low, so that the camera pose can be reliably estimated even in very challenging situations. Experimental results demonstrate that the proposed system can robustly work in dynamic environments and outperforms the state-of-the-art SLAM systems (e.g. PTAM).

Index Terms: I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Tracking; H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities

1 Introduction

Camera tracking is the key component of robot navigation and augmented reality applications, which provides the necessary camera parameters to guild the route, or set the virtual cameras of the 3D environment. Many of the previous works require the prior knowledge of the target scene, such as point cloud [9], CAD model [30], or artificial markers [12], etc. These model-based camera tracking techniques generally require accurate 3D models [19], which greatly limits their usage in practice since the complex natural scenes are usually difficult to reconstruct.

In recent years, Davison et al. [8] propose a real-time camera tracking system called monoSLAM (monocular Simultaneously Localization and Mapping), which can recover the 3D structure of the unprepared scenes at the same time. Since monoSLAM uses an extended Kalman filter to solve the 3D points and camera poses simultaneously, it can be considered as a filter-based SLAM. Klein and Murray [17] propose to solve the SLAM problem with a multi-threading framework based on a structure from motion (SfM) technique. They assign the computation burdens of camera tracking and scene mapping onto two working threads, and achieve a real-time SfM system (called PTAM) in a parallel framework. Strasdat et al. [36] further compare SfM-based SLAM with filter-based SLAM, and conclude that the former outperforms both in precision and scalability. Since both monoSLAM and PTAM are based on 3D sparse features of the scene, more recent works, such as DTAM [25], propose to track the camera motion by direct per-pixel alignment between the color image and inverse depth image, which is proved more resilient to degraded images and rapid motion than the sparse features based methods.

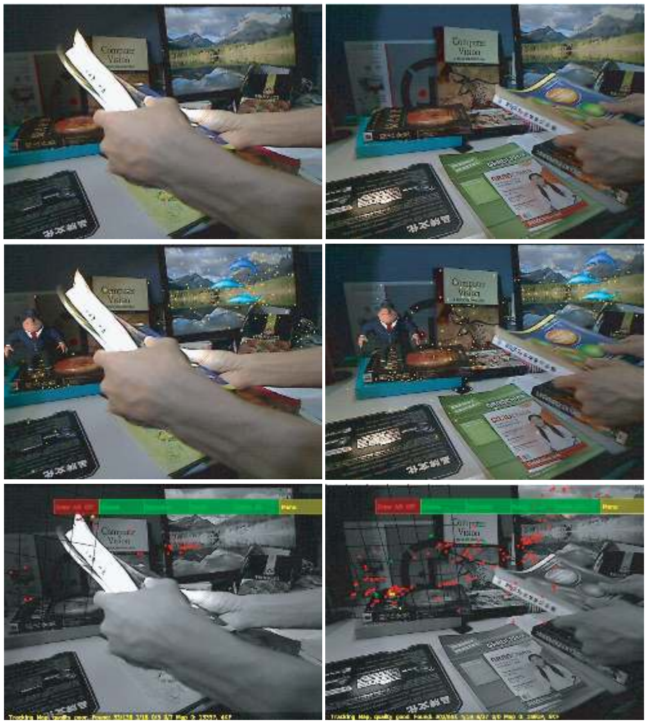

Figure 1: A challenging example. Top: two selected source frames. Middle: the SLAM result (with augmented reality) of our system. Bottom: the SLAM result of PTAM [17, 18].

However, all above SLAM systems have their own shortcomings. The most significant problem is that the target scene must keep stationary during processing, since they did not consider updating the 3D structure corresponding to dynamic objects including illumination changes, otherwise the recovered 3D models will become obsolete rapidly, and the system has to start from scratch. Most recently, 3D depth cameras (e.g. Microsoft Kinect) are becoming more and more popular in camera tracking. The KinectFusion system [24] fuses the captured depth maps into a scene model maintained with a volumetric, truncated signed distance function representation, and uses multi-scale iterative closest point alignment between the predicted surface and current captured depth map to track camera motion. It is notable that even when the target scenes are gradually changing during the process, the system can adjust the 3D structure accordingly by the depth fusion, and the tracking process will continue with the dynamic structure.

The KinectFusion system heavily relies on the Kinect's 3D depth camera, whilst it is still an open problem by merely using normal color cameras, despite many existing works on localization in dynamic environments [14, 1, 28, 16, 43]. In this paper, we propose a novel robust invariant feature based SLAM system which can deal with the challenging cases where the scene contains large moving

*e-mails: {zhangguofeng,bao}@cad.zju.edu.cn

objects or gradually changes. Our framework is similar with Pirker et al. [28]'s CD SLAM, which can achieve tracking and mapping in a dynamic world, but they mainly use the visibility information of reference 3D features to filter out potential ambiguities in the 2D-3D matching process, which is based on the HoC [27] descriptor. On the contrary, we propose to discard the obsolete features and keyframes detected by comparing feature appearance and structure, and our system can robustly handle occlusions.

Our framework is similar to PTAM, but we employ the invariant features (i.e. SIFT [20]) instead of FAST [32] to perform global matching, which is more robust to fast movement and camera relocalization, and can be easily extended to handle a larger scale scene. Furthermore, we contribute a novel prior-based adaptive RANSAC algorithm with online 3D points and keyframe updating scheme so that the proposed system can adaptively model the dynamic environments which may have significant illumination or structure changes. Especially, the occlusions can be reliably detected and handled. Figure 1 shows a challenging example, where the scene is changing gradually with significant illumination variation. Our system can robustly recover the camera motion, but PTAM fails. Due to expensive SIFT feature extraction and matching, our system currently requires GPU acceleration with multi-thread programming to achieve real-time performance (25fps) in a desktop PC with a 4-core CPU.

2 Related Work

There have been a great amount of literature on camera pose estimation and 3D reconstruction. We will briefly review the most relevant works on SLAM systems, along with some important extensions to the original version.

2.1 Filter-based SLAM

MonoSLAM [8] is the pioneering work on real-time camera tracking while mapping the unprepared scenes. Their system is initialized by detecting an artificial marker with known size in the scene. The positions of all landmarks and the camera pose are intimately linked in a probabilistic filtering framework of joint states based on extended Kalman filter, and updated together with every input frame. MonoSLAM tracks the features in an active search manner with correlating templates under a predicted motion model, which is not invariant to viewing direction and easily causes ambiguities in cluttered environments, so a simplified SIFT algorithm [4] is then employed to alleviate this problem. Despite the high efficiency of EKF, the computation complexity of monoSLAM is $O(N^2)$, where N is the number of landmarks. So the number of landmarks is generally limited to be a few hundreds, and the scalability is restricted to a small space if without using a delicate feature selection method [33]. Eade and Drummond [10] present a monocular SLAM system that employs a FastSLAM-type [21] particle filter and top-down search to allow real-time performance while mapping hundreds of landmarks.

2.2 SfM-based SLAM

The structure from motion (SfM) technique provides a fundamental framework for offline 3D structure reconstruction from photo collections [35] or video sequences [41]. The core algorithm, i.e. bundle adjustment (BA), is generally considered too computationally expensive to be invoked in real-time application. In recent years, lots of optimization methods [11, 40] have been proposed to accelerate BA, where local bundle adjustment involving tens of keyframes and hundreds of 3D points has been proved to be solved at interactive rate [22]. With online BA, Klein and Murray [17] propose a parallel tracking and mapping (PTAM) framework, which is actually a real-time SfM system. PTAM splits mapping and tracking into two separate tasks, running in two parallel threads. The former deals with adding new 3D points and keyframes into the 3D map, and optimizing the local structures with local BA. The tracking thread will match the 2D-3D features similar to monoSLAM, and estimate the camera pose for each input frame in real-time. To improve the agility of PTAM, they also add edge features to the map to enhance the tracking under fast motion, and an inter-frame rotation estimator to aid tracking under rapid panning [18]. They further propose a relocalization method using the fuzzy representation of the existing dense keyframes, which enables switching among multiple scenes [3]. Our work also belongs to SfM-based SLAM and achieves the above objectives in a more unified way thanks to the use of SIFT features with keyframe representation.

2.3 SLAM in Dynamic Environments

To deal with a dynamic scene, it is necessary to identify the dynamic contents from the static parts. Hahnel et al. [14] use Expectation-Maximization (EM) algorithm to update the probabilistic estimate about which measurement (obtained with a laser-range scanner) might correspond to a static/dynamic object. Bibby and Reid [1]'s SLAMIDE also estimates the state of 3D features (stationary or dynamic) with a generalised EM algorithm. They use reversible model selection to include dynamic objects into SLAM, so that the system can include the dynamic objects in a single framework. If the scene changes frequently as in our examples, the resulting map may contain many unnecessary 3D features, which is obviously quite a waste of memory and computation, and also may suffer from robustness problems. Bleser et al. [2] propose a CAD model based SLAM system similar to monoSLAM [8], which will delete features from the map if they have not been tracked in more than 50 percent of the frames where they should be visible (predicted by camera poses). In their system, the 3D features are triangulated separately, so the overall structure is less precise than SfM-based SLAM, such as PTAM. Under the framework of PTAM, Shimamura et al. [34]'s vSLAM estimates the flow vector throughout all the outliers after camera pose estimation on the tracking thread, and clusters the flow vectors by GMM. If the number of outliers in a cluster exceeds a threshold, the outliers are considered to be upon moving objects and eliminated from the map. Both systems can keep the map tight and precise, but in case of large occlusions (e.g. large moving objects appear), a standard RANSAC process may easily fail to accurately recover the camera pose. In addition, vSLAM does not remove any keyframes, so the 3D features of new objects covered by existing keyframes cannot be added. Our system can address these two problems by updating keyframes with 3D points and using a new prior-based adaptive RANSAC algorithm.

With multiple cameras, we can not only detect the dynamic contents, but also recover the trajectories of moving objects. Imre et al. [15] use a set of fixed calibrated cameras to recover the reference 3D structure of the scene, which will support the tracking of other moving cameras. In Zou and Tan [43]'s CoSLAM system, all cameras can move freely in the scene, where each camera works independently with intra-camera pose estimation, and both static and dynamic points are used to obtain inter-camera pose estimation for all cameras.

2.4 3D Change Detection

The scene change detection is very common in 2D scenarios, such as background substraction, surveillance, medical diagnosis etc. [31]. Since these approaches are sensitive to the changes of illumination and viewpoints, they are not appropriate to be directly used in 3D scenes. Pollard and Mundy [29] represent the target 3D space in a bounded volume partitioned into voxels, and each voxel contains one surface probability and one color model. Then the initial photo collections along with their solved camera poses are fed to the system one by one to update the probabilities of the voxels spanned by the ray from the camera center until the system converges. As a result, to determine whether a pixel has changed is

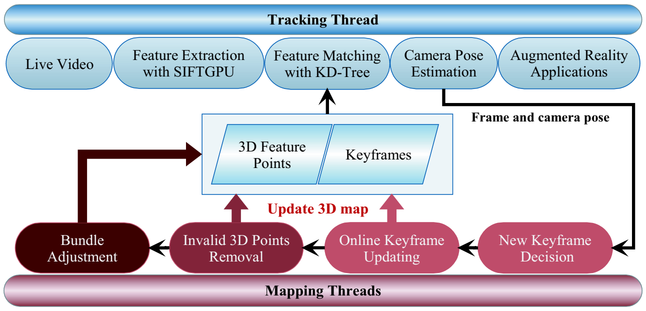

Figure 2: Our framework.

simply to compute the probability of seeing the pixel color from the trained color models. Taneja et al. [37] argue that changes in image appearance may not lead to changes in the geometry, and propose a graph based method. They divide the old dense 3D models into a set of voxels, which correspond to the graph nodes. For the binary term on edges, they observe that the neighboring voxels with similar colors may change together. For the unary term on nodes, the voxels are reprojected into all the visible input images, and the pixel color difference between each pair of projections are computed. The formulated Gibbs energy function is solved by Graph Cuts algorithm. For robust real-time SLAM, our objective is to detect the 3D points whose positions or appearances have changed, which is similar to [37], however we only have a bunch of sparse 3D points and keyframes at hands, so the voxel based solutions are not applicable in our situation.

3 Framework

We first give an overview of our system, as illustrated in Figure 2. We assume the intrinsic camera matrix K is known and constant. Similar to PTAM [17, 18], we simultaneously estimate the scene structure and camera motion (6 DOF) without any prior knowledge of the scene. Different to PTAM, we extract SIFT features for each online frame and match them between adjacent frames, where SiftGPU [39] is employed in our implementation. Similar to [9], we use keyframes to represent the scene. The keyframe selection and updating algorithm will be elaborated in Section 4.

We first initialize the structure and motion with two initially selected keyframes by the method of [26]. The feature points on the KD-Tree are defined as reference features, whose 3D positions are already estimated. We use KD-Tree to accelerate the matching process. The 3D map is built gradually along with the added keyframes. If a new keyframe is added, more 3D features will be reconstructed and added into the map, so the KD-Tree should be updated accordingly. Because the construction of KD-Tree is time consuming when the number of 3D points is large, our system maintains two KD-Trees, where one is active for real-time matching, and the other is waiting for updating when the number of new 3D features from the reconstruction thread exceeds the threshold. For each online frame, we extract the SIFT features and match them with the active KD-Tree. If the background KD-Tree has been updated, it will become active, and the original active KD-Tree will change to wait for update in the next round.

For the newly extracted SIFT features in the current frame, we match them with the reference features by comparing their descriptors. For feature point $\mathbf{x}$, its nearest two descriptors in the KD-Tree are denoted as $N_1(\mathbf{x})$ and $N_2(\mathbf{x})$, respectively. Similar to [20], we define the matching confidence between $\mathbf{x}$ and $N_1(\mathbf{x})$ as

$$c = \frac{||\mathbf{p}(\mathbf{x}) - \mathbf{p}(N_1(\mathbf{x}))||}{||\mathbf{p}(\mathbf{x}) - \mathbf{p}(N_2(\mathbf{x}))||},$$

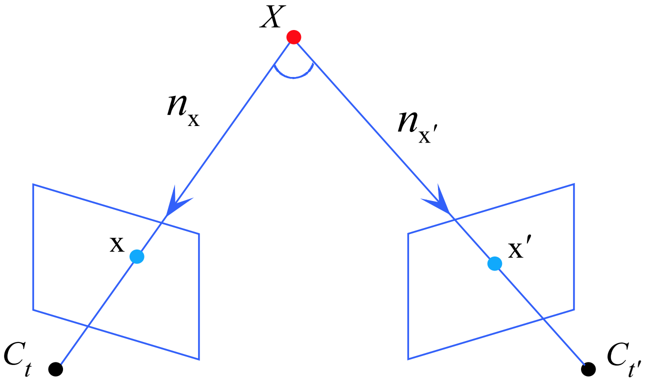

Figure 3: Viewing direction illustration.

where $\mathbf{p}(\mathbf{x})$ denotes the descriptor of feature $\mathbf{x}$. If $c < 0.6$, we assume $\mathbf{x}$ and $N_1(\mathbf{x})$ are matched.

Since the 3D positions of the feature points in KD-Tree have already been estimated, with a set of 2D-3D correspondences, we can quickly estimate the camera pose (i.e. R and T) by minimizing

$$\min_{R,t} \sum_i ||\mathbf{x}_i - \pi(K(RX_i + T))||^2, \qquad (1)$$

where $\mathbf{x}_i$ is the ith 2D feature, and X_i is its corresponding 3D point. π is a projection function. Similar to [17], we also run a work thread for frequent bundle adjustment of the estimated 3D points and camera poses.

The above strategy with a simple online keyframe selection method (e.g. [17]) generally works rather well for a static scene. However, if the scene contains large moving objects or gradually changes, some reference features and selected keyframes will be invalid and need to be replaced in time. In the following section, we will introduce a novel online keyframe and 3D points updating scheme to effectively address this problem.

4 Online 3D Points and Keyframe Updating

For each input frame, if it satisfies the following three conditions, it will be selected as a keyframe: 1) the camera pose is successfully estimated, 2) has fewer than m_1 ($m_1 = 80$ in our experiments) existing common features with the existing keyframes, 3) contributes more than m_2 new 3D points (m_2 is generally set to $50 \sim 100$ in our experiments), which are generated by feature matching and triangulation between input frame and existing keyframes.

If the input frame already satisfies the previous two conditions, we will consider it as a potential keyframe, and then select 5 most related keyframes that have maximum common features with it. We perform feature matching again between the input fame and 5 selected keyframes. The features in the input frame that already have found correspondences do not need to be matched again. Similar to [9], we construct a KD-Tree for each keyframe to speedup the matching. Epipolar geometry constraint [42] is also used to reject outliers and improve the matching accuracy. The matched points among the input frame and keyframes are recorded to constitute a feature track. For each track $\mathscr{X}_i$, we estimate its 3D position by

$$X_i = \min_{X_i} \sum_{j \in \phi(\mathscr{X}_i)} ||\mathbf{x}_{ij} - \pi(K(R_j X_i + T_j))||^2,$$

where $\phi(\mathscr{X}_i)$ denotes the set of frames that $\mathscr{X}_i$ appears in. If the number of newly estimated 3D points satisfies the third condition (i.e. larger than m_2), we will select the input frame as a keyframe and add it to the keyframes set. These newly estimated

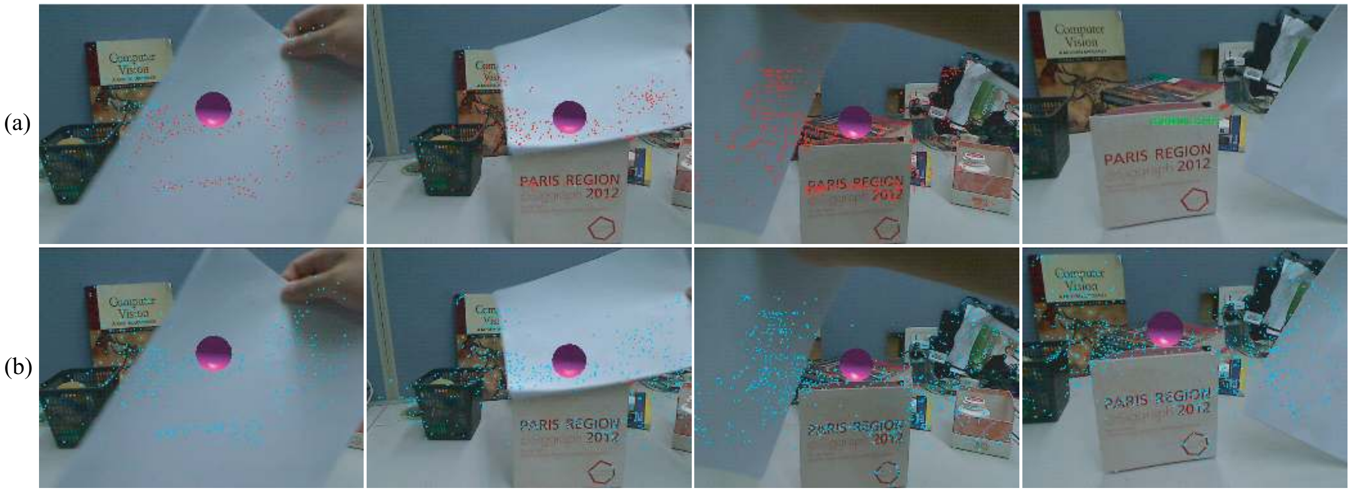

Figure 4: Handling occlusions caused by dynamic objects. (a)The SLAM result without occlusion handling. The valid 3D points are highlighted with cyan color. Since most 3D points are recognized as "invalid" 3D points (highlighted with red color) due to occlusions caused by a moving paper, the estimated camera may frequently drift or even lost. (b) The SLAM result with occlusion handling. Almost all the occluded points are successfully identified and not set to invalid, so the camera poses can be reliably estimated.

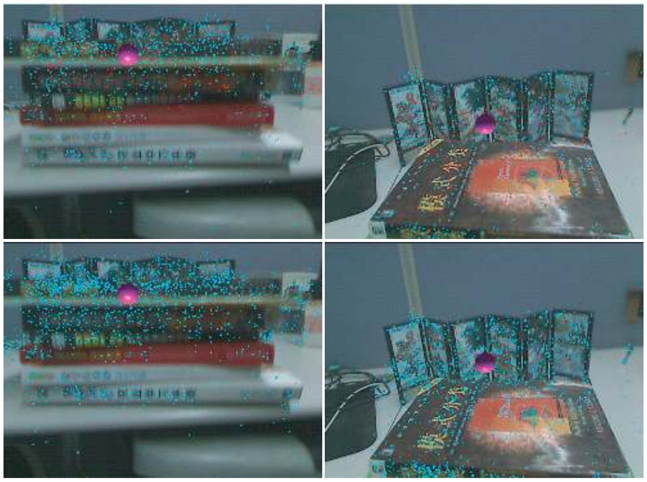

Figure 5: Handling occlusions caused by static objects. Top: the SLAM result without occlusion handling. The valid 3D points are highlighted with cyan color. Many 3D points are recognized as "invalid" 3D points (highlighted with red color) due to occlusions. Bottom: the SLAM result with occlusion handling. Most of the occluded points are successfully identified and not set to invalid.

3D points are also selected as reference features along with the new keyframe. The KD-Tree of the whole 3D map is updated accordingly.

For each input frame, we also need to measure whether parts of the scene have changed. Specifically, we select 5 closest keyframes by comparing the rotational matrices and translational vectors. Firstly, we compare the direction of Z axis between current frame and previous frames. We select the frames whose direction differences to current frame are less than a threshold, and then further compare their camera positions to determine the 5 closest frames. Then for each selected keyframe, we first compare the color histogram with current frame for the R, G, B channels respectively. Each bin value is normalized as $\sum_{i=0}^{255} c_i^2 = 1$, where c_i is the normalized bin value for R/G/B channel at ith bin. We define the difference of color histogram as a sum of distance among the three histograms. If the difference is larger than 1, it is very likely that some areas may have changed so that we should project the feature points in this keyframe to the current frame. For feature point $\mathbf{x}$ in keyframe, its 3D position is denoted as X and its projection in the current frame is denoted as $\mathbf{x}'$. We use $V(X)$ to denote the status of 3D point X. If X is invalid, we set $V(X) = 0$, otherwise $V(X) = 1$. Then we compare the appearance and 3D structure between $\mathbf{x}$ and $\mathbf{x}'$ to determine whether X is valid or invalid.

The viewing directions from X to the camera centers of the keyframe and current frame are denoted as $n_{\mathbf{x}}$ and $n_{\mathbf{x}'}$, respectively. Figure 3 gives an illustration. The angle between $n_{\mathbf{x}}$ and $n_{\mathbf{x}'}$ can be computed as $\arccos(n_{\mathbf{x}}^\top n_{\mathbf{x}'})$. If $n_{\mathbf{x}}^\top n_{\mathbf{x}'} < \tau_n$ (τ_n is generally set to $\cos(30^\circ)$ in our experiments), feature point $\mathbf{x}$ is very likely to be invisible in the current frame or there is large perspective distortion (e.g. the camera is rotating 180° around an object), so it is better to keep $V(X) = 1$. Otherwise, we will further compare the appearance difference between $\mathbf{x}$ and $\mathbf{x}'$ as:

$$D_c(X) = \min_d \sum_{\mathbf{y} \in W(\mathbf{x})} |I_{\mathbf{y}} - I_{\mathbf{y}'+d}|, \tag{2}$$

where $W(\mathbf{x})$ denotes the window centered at $\mathbf{x}$, and is set to 11×11 in our experiments. $I_{\mathbf{y}}$ denotes the color of $\mathbf{y}$. $\mathbf{y}'$ is the projection of $\mathbf{y}$ by the estimated depth and camera parameters. d is a small translational vector. Ideally, $\mathbf{y}'$ should be exactly the correspondence of $\mathbf{y}$. However, due to estimation error, $\mathbf{y}'$ may deviate from the true position. Therefore, we involve a local search with displacement d in (2) to alleviate this problem. If $D_c(X) > \tau_c$, it is very likely that the appearance of $\mathbf{x}$ has changed or occlusion occurs. For example, the region may be temporarily occluded by a moving object or even a static object due to viewpoint change. If the appearance of $\mathbf{x}$ has changed due to illumination or position variation, we should set $V(X) = 0$. Otherwise we should not remove X. So if we directly set $V(X) = 0$ when $D_c(X) > \tau_c$, many temporarily occluded points will be removed quickly which may cause drift or camera lost problem as shown in Figure 4(a) and our supplementary video. Therefore we need to detect the occlusion to avoid this problem, which is described in Algorithm 1.

Denote $\phi(\mathbf{x}')$ as the set of tracked feature points in the current frame whose distance to $\mathbf{x}'$ is less than r_1 pixels (r_1 is generally set to 20 in our experiments, except for the example in Figure 9 where

Algorithm 1 3D Points Updating with Occlusion Handling

```
for each valid feature point $\mathbf{x}$ (its corresponding 3D point is $X$) in each selected keyframe, do
  Compute its projection $\mathbf{x}'$ in the current frame.
  if $n_{\mathbf{x}}^{\top} \hat{n}_{\mathbf{x}'} < \tau_n$, then
    compute the appearance difference $D_c(X)$ by (2).
    if $D_c(X) > \tau_c$, then
      find a set of tracked feature points $\phi(\mathbf{x}')$ in the current frame whose distance to $\mathbf{x}'$ is less than $r_1$ pixels.
      if $\phi(\mathbf{x}')$ is not empty and $z_{X_{\mathbf{y}}} \geq z_X$ for all $\mathbf{y} \in \phi(\mathbf{x}')$, then
        set $V(X) = 0$.
      else
        for each point $\mathbf{y} \in \phi(\mathbf{x}')$, do
          project $X$ and $X_{\mathbf{y}}$ to the frame where $X$ initially appeared.
          if $|\mathbf{x}_p - \mathbf{y}_p| < r_2$, then
            set $V(X) = 0$.
          end if
        end for
      end if
    end if
  end if
end for
```

$r_1 = 40$.). If X is occluded by a moving object, $\phi(\mathbf{x}')$ should be empty. The reason is that dynamic points violate the multi-view constraint and will be recognized as outliers. Thus there are generally no tracked 3D points in a moving object. In this case, X should not be set to invalid. Excluding this case, $\phi(\mathbf{x}')$ generally contains at least one point $\mathbf{y}$. We denote the corresponding 3D point of $\mathbf{y} \in \phi(\mathbf{x}')$ as $X_{\mathbf{y}}$. We compare the depth values between X and $X_{\mathbf{y}}$. If $z_{X_{\mathbf{y}}} \geq z_X$ for all $\mathbf{y} \in \phi(\mathbf{x}')$, where $z_{X_{\mathbf{y}}}$ and z_X are their respective depth values, it is safe to set $V(X) = 0$ because nothing occludes X so X must have changed. Otherwise, for those $X_{\mathbf{y}}$ with $z_{X_{\mathbf{y}}} < z_X$, there are two cases: 1) $X_{\mathbf{y}}$ appears due to the change of X (illumination or position variation); 2) $X_{\mathbf{y}}$ belongs to an existing static object which temporarily occludes X due to viewpoint change. We need to set $V(X) = 0$ if and only if it is the first case. Actually, we can distinguish these two cases by projecting X and $X_{\mathbf{y}}$ to the keyframe where X initially appeared. Their projections are denoted as $\mathbf{x}_p$ and $\mathbf{y}_p$, respectively. In the first case, $X_{\mathbf{y}}$ appears due to the change of X, and is very likely to be on the same object with X. So $X_{\mathbf{y}}$ is generally close to X, i.e. $|\mathbf{x}_p - \mathbf{y}_p|$ is small. In the second case, $|\mathbf{x}_p - \mathbf{y}_p|$ is generally not small because X was not occluded by $X_{\mathbf{y}}$ when X initially appeared, otherwise X cannot be extracted. Based on this observation, we generally can reliably distinguish the above two cases by checking $|\mathbf{x}_p - \mathbf{y}_p|$. If there is at least one point $\mathbf{y} \in \phi(\mathbf{x}')$ satisfying $|\mathbf{x}_p - \mathbf{y}_p| < r_2$ (r_2 is generally set to 20 in our experiments), we will set $V(X) = 0$; otherwise, we keep $V(X) = 1$. With this strategy, we can reliably detect the changed 3D points and alleviate the effect of occlusions, as illustrated in Figures 4 and 5. Although a few occluded 3D features may still be recognized as invalid 3D points, it generally does not harm camera tracking since our system will quickly add new 3D features in time.

For each keyframe, if the number of invalid 3D points and the common features shared with other keyframes is larger than 90% of its total point number, we will label this keyframe as invalid.

5 Prior-based Adaptive RANSAC

If the scene contains too many dynamic feature points, especially when the dynamic points also undergo a rigid motion, it may be not reliable using the standard RANSAC [13], since the standard RANSAC always selects the motion with the maximal number of supporting points. Here, we propose a prior-based adaptive RANSAC algorithm (PARSAC) to address this problem. The basic flow is similar to that of the standard RANSAC. In the first step, a hypothesis is generated by randomly sampling a minimal subset of the input data, and then computing the model parameters fitting this minimal sample. In the second step, the hypothesis is evaluated using all points, and the best hypothesis is recorded. These two steps are repeated until a certain termination criterion is satisfied. We have a key observation that the static background features usually distribute evenly, whereas the dynamic foreground points may aggregate in a few small textured areas. For robust camera pose estimation, the matches are required to be not too close or lie on a line. Therefore, not only the number but also the distribution of inliers should be taken into account while determining the best hypothesis.

Sample generation. Myatt et al. [23] assume that inliers are close together and propose to sample each minimal subset based on proximity. In our scenario, since only dynamic points are close together, proximity based sampling will results in many minimal subsets containing only the undesired dynamic points. We propose to avoid this situation by enforcing the sampled points distributed as even as possible in the whole image. Specifically, we evenly divide the whole image to 10×10 bins. For each bin B_i, we only select one point at most. This strategy can avoid the sampled points being gathered in a small area. We can sample bins randomly. However, if the inlier ratio is very low, a pure random sampling strategy may be rather time-consuming to generate a good hypothesis. Actually, we can use the inlier/outlier distribution information from the previous frame to guide the point sampling of the current frame since the image content among adjacent frames is rather similar. Specifically, we record the inlier ratio for each bin in the previous frame, and truncate it to $[0.2, 1]$ to guarantee that each bin could have a certain chance to be sampled for the current frame. The truncated inlier ratio of B_i for the previous frame is denoted as ε_i^*. Then for bin sampling in the current frame, we define the probability distribution as follows:

$$p_i = \varepsilon_i^* / \sum_j \varepsilon_j^*.$$

p_i denotes the probability of sampling a point from B_i. We sample the points from bins according to the estimated probability distribution.

Hypothesis evaluation. We evaluate a hypothesis not only by the number of supporting points, but also by their distributions. Given a hypothesis, we compute the reprojection error for all points to determine the inliers and outliers, so that the distribution of inliers can be described by a covariance matrix

$$C = \frac{1}{N-1} \sum_i (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T, \qquad (3)$$

where N is the number of inliers, $\mathbf{x}_i$ is the image location of ith inlier, and $\bar{\mathbf{x}} = \frac{1}{N} \sum_i \mathbf{x}_i$ is the mean location of all inliers. Here we make a slight modification by taking into account the inlier ratio. We use bin center to represent all inliers lying within the bin. For each bin B_i, its confidence weight is set to its inlier ratio ε_i. Then we replace (3) with the following weighted covariance matrix

$$C = \frac{\sum_i \varepsilon_i}{(\sum_i \varepsilon_i)^2 - \sum_i \varepsilon_i^2} \sum_i \varepsilon_i (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T, \qquad (4)$$

where $\mathbf{x}_i$ is the center of B_i, and $\bar{\mathbf{x}}$ is the weighted mean position computed as

$$\bar{\mathbf{x}} = \frac{1}{\sum_i \varepsilon_i} \sum_i \varepsilon_i \mathbf{x}_i.$$

Equation (4) can be considered as the generalization of (3). If we only pick bins having inliers to construct the covariance matrix, and

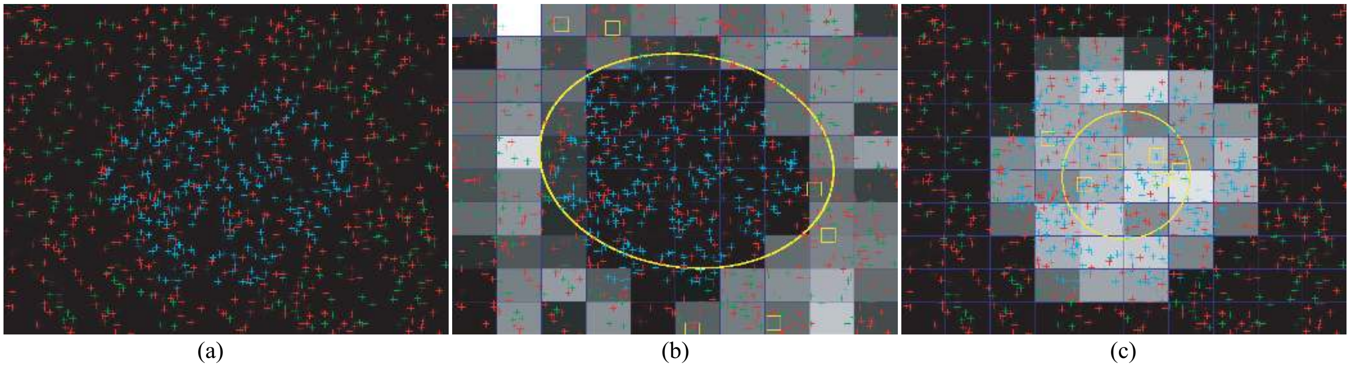

Figure 6: The synthetic example for outlier rejection evaluation. (a) The synthetic scene contains three kinds of points: 200 green points are on the static background, 300 cyan points on the rigidly moving object, and other 500 red points are random moving. (b) A hypothesis with a sample containing only green points. (c) A hypothesis with a sample containing only cyan points. The image is divided into 10×10 bins. The brightness of each bin represents its inlier ratio. The sampled points are highlighted with small rectangles. The yellow ellipse denotes the computed covariance matrix of the sample.

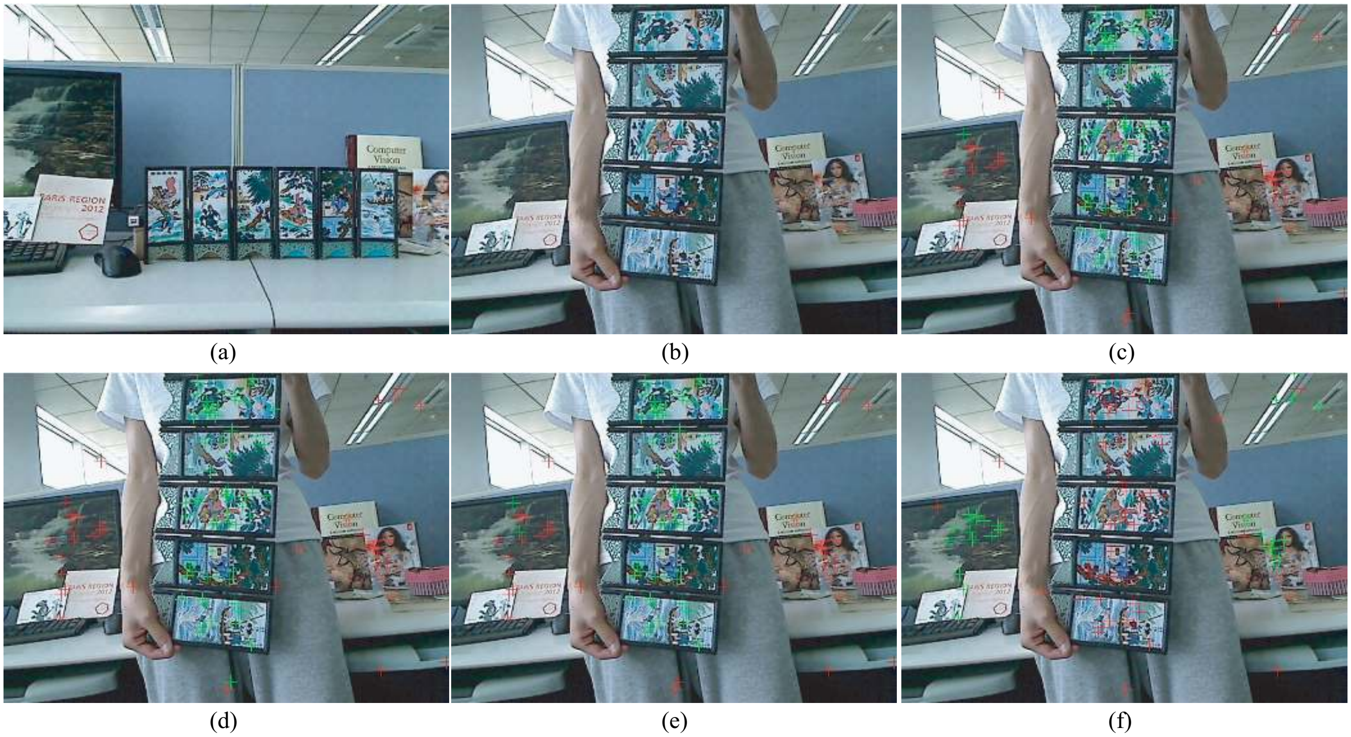

Figure 7: A real example for outlier rejection. (a-b) Two real images. (c) The recognized inlier points by RANSAC [13], marked with green crosses. The outliers are marked with red crosses. (d) The recognized inlier points by LoSAC [7]. (e) The recognized inlier points by ProSAC [6]. (f) The recognized inlier points by our ARSAC method.

treat each inlier bin equally (set $\varepsilon = 1/N$, where N is the number of inlier bins), then (4) will become the standard covariance matrix as in (3). With a weighting scheme, the problem that some outlier matches are coincidentally consistent with the evaluated hypothesis can be reduced, since the weights of these corresponding bins will be rather small. The influence from the bins that contain both static and moving points also can be alleviated. Finally, we score the evaluated hypothesis as

$$s = (\sum_i \varepsilon_i) \frac{\pi\sqrt{det(C)}}{A}, \tag{5}$$

where A is the image size for normalization. $\sum_i \varepsilon_i$ represents the weighted number of inliers, and $\pi\sqrt{det(C)}$ is the area size of the ellipse representing the inlier distribution. The intuition is that the best hypothesis should have a large number of inliers which are evenly distributed in the whole image.

Termination criterion. If the sampling probability distribution is accurate, a good sample without outliers can be quickly obtained with a few iterations. Otherwise, the sampling process should be repeated until the probability of finding a better hypothesis is lower than a threshold. We model this probability by assessing how well the sampling probability distribution fits the currently best hypothesis. Denote $\hat{\varepsilon}_i$ as the inlier ratio of B_i for the currently best hypoth-

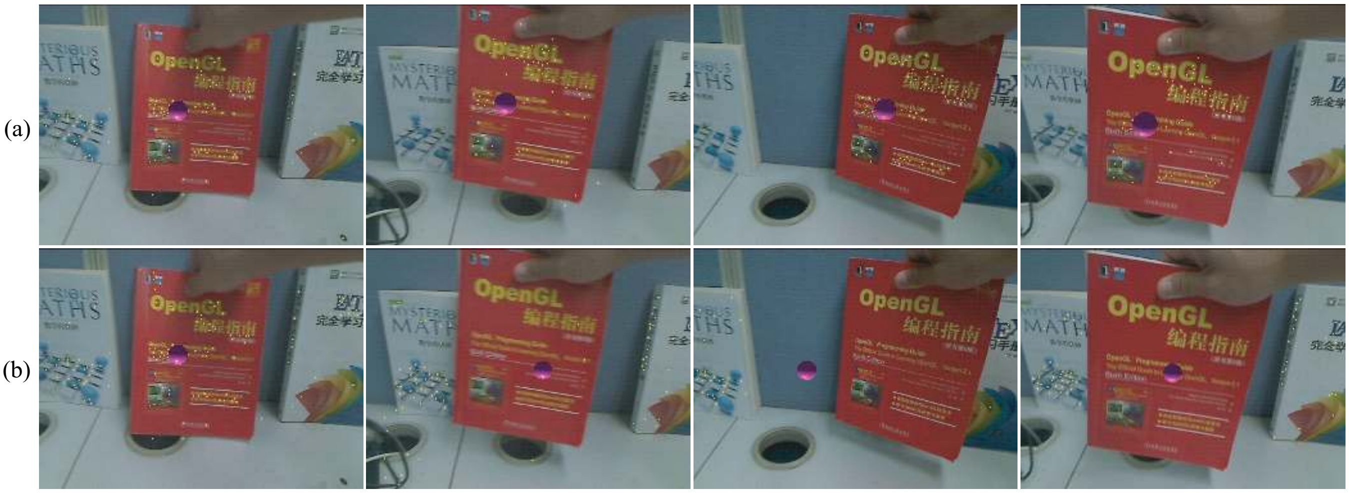

Figure 8: SLAM result comparison with different robust estimators. (a) The SLAM result with standard RANSAC. The inserted purple sphere drifts with the moving red book, which is not accurate. (b) The SLAM result with our PARSAC. The inserted purple sphere stays at accurate position without drift.

Table 1: The average running time of different algorithms.

	RANSAC	LoSAC	Multi-GS	PARSAC	ARSAC
Running Time	15.2ms	18.4ms	92.7ms	20.5ms	19.3ms
Iterations	500	500	500	500	500

Table 2: The success ratio of different algorithms.

Iterations	RANSAC	LoSAC	Multi-GS	PARSAC	ARSAC
500	12%	8%	100% / 57%	65%	6%
1,000	15%	7%	100% / 63%	95%	14%
2,000	7%	8%	100% / 56%	100%	30%
5,000	5%	4%	100% / 52%	100%	66%
10,000	0%	5%	100% / 48%	100%	83%
20,000	0%	7%	100% / 53%	100%	96%

esis, the probability of selecting an inlier point is $\sum_i p_i\hat{\varepsilon}_i$. The probability of selecting a sample without outliers is $(\sum_i p_i\hat{\varepsilon}_i)^m$, where m is the sample size. For projective matrix estimation, $m = 6$. Then the number of the iterations K_s must satisfy the following termination criterion

$$(1-(\sum_i p_i\hat{\varepsilon}_i)^m)^{K_s} < \eta.$$

That is to say if the probability that the K_s samples all contain outliers is less than η, we will stop the iteration.

After finding the best sample, the projective matrix is further refined with all inliers. Finally, we decompose the estimated projective matrix to R and T, and then refine them by minimizing (1).

There are already several robust estimation algorithms [13, 38, 7, 6, 23, 5] have been proposed. Here, we compared our outlier rejection algorithm with the standard RANSAC [13], LoSAC [7], and Multi-GS [5]. In order to verify the effectiveness of the inlier ratio prior, we also define another algorithm ARSAC by removing the inlier ratio prior from PARSAC. We synthesize a scene that contains a static background and a rigidly moving object. The camera and the rigid object undergo different motion. As shown in Figure 6(a), the synthetic scene contains 200 static points, 300 dynamic points in a rigidly moving object, and other 500 randomly moving points. Each point is contaminated with a standard Gaussian noise. We synthesize 100 frames in total. We found that both RANSAC and LoSAC fail to recognize the static points, thus fail to accurately estimate the camera motion, since the number of static points is smaller than the number of rigidly moving points. Multi-GS generally can distinguish static and dynamic points since it explicitly detects multiple hypotheses. However, it cannot determine which hypothesis is better. In contrast, our method not only detects these two motions, but also recognizes the desired camera motion. In addition, the computational overheads of both PARSAC and ARSAC are much smaller than Multi-GS, as listed in Table 1 (the iteration number is set to 500). It should be noted that although the computational overhead of PARSAC is slightly larger than that of ARSAC for each iteration, PARSAC can quickly find the best hypothesis with much fewer iterations. Table 2 shows the success ratio (for all 100 frames) of different algorithms along with the growing iteration number. For this challenging data (the inlier ratio is only 20%), PARSAC already can achieve 95% success ratio with $1,000$ iterations thanks to the use of inlier ratio prior, but ARSAC needs $20,000$ iterations to achieve comparable success ratio. We also found that the success ratio of RANSAC even becomes smaller when the iteration number increases. The reason is that the number of static points is smaller than the number of rigidly moving points, so RANSAC will favor to recognize the rigidly moving points as inliers. It should be noted that Multi-GS actually often only recognizes static points in this example (the success ratio of recognizing the rigidly moving points is only around 60%), although it is supposed to accurately recognize both two kinds of hypothesis. The reason is that there are too many outliers, which make the preference analysis of Multi-GS not reliable. If the outlier ratio is small, Multi-GS generally works well.

Figure 6(b) and (c) show two different hypothesis by our method (one sample contains only static points, and one sample contains only rigidly moving points), where the computed corresponding $\sum_i \varepsilon_i$ are quite comparable, 24.94 and 21.77, respectively. Taking into account the covariance matrix of the sample by (5), we have $s_1 = 8.31 > s_2 = 1.89$, so a better hypothesis is accurately determined.

Figure 7 shows an example with two real images captured from different viewpoints. As can be seen, the five-leaf screen undergoes a large rigid movement. We estimate the fundamental matrix between these two images by different algorithms to remove outliers. ProSAC [6] algorithm needs a prior ordering of all data points depending on the quality of each feature match. Here we sort all the feature matches by Euclidean distance of SIFT descriptor in ascending order. The recognized inlier points (marked with green crosses) by RANSAC, LoSAC and ProSAC are all on the five-leaf screen,

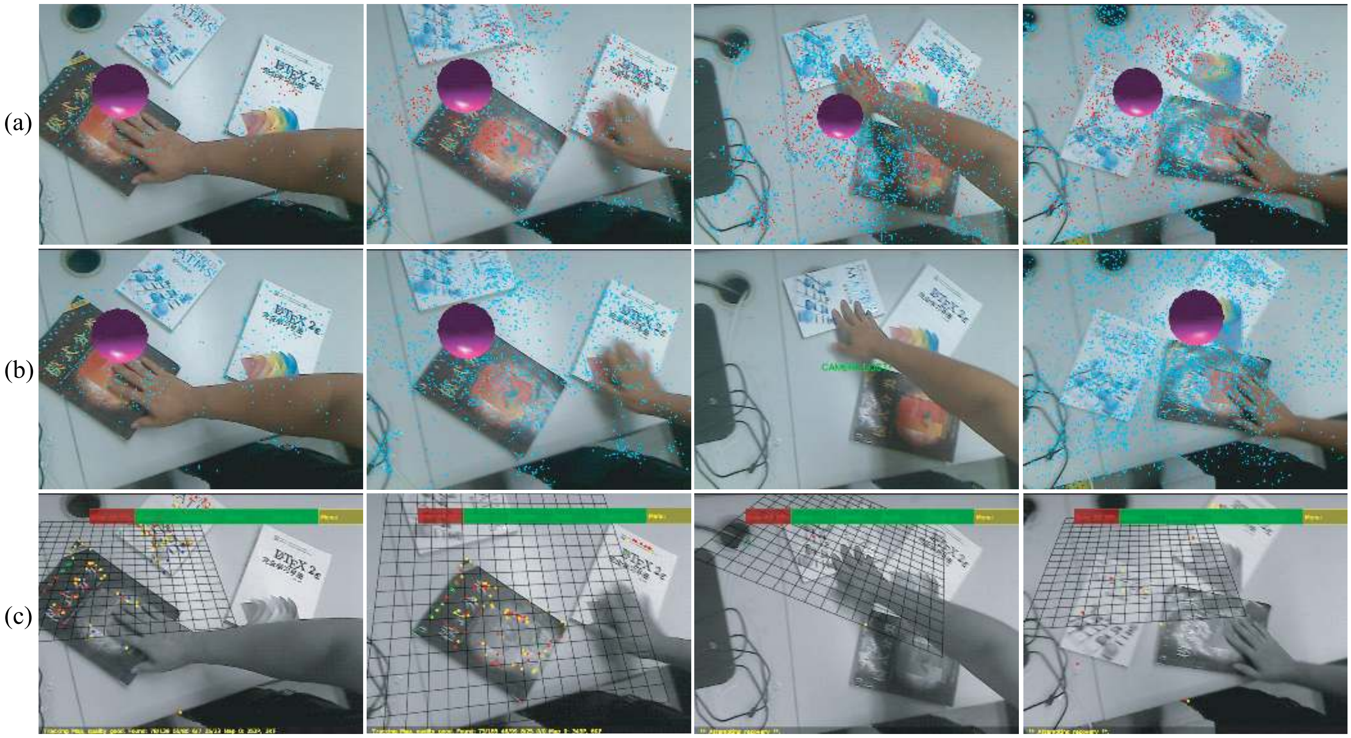

Figure 9: SLAM result comparison. (a) Our SLAM result with the component of removing invalid 3D points and keyframes. The changed 3D points are detected and highlighted with red color. (b) Our SLAM result without removing invalid 3D points and keyframes. (c) The SLAM result of PTAM.

which is not correct. In contrast, our ARSAC method can correctly recognize the static points.

Figures 8(a) and (b) show the SLAM results of the same system but with different robust estimators (i.e. standard RANSAC and our PARSAC, respectively). As can be seen, standard RANSAC recognizes the matched features on the moving red book as inliers in many frames, so the inserted purple sphere drifts with the moving book. In contrast, our PARSAC can accurately recognize the inliers/outliers, so the inserted sphere can stay at accurate position without drift. Please refer to our supplementary video[1] for the complete frames and comparison.

6 Experimental Results

Table 3: Processing time per frame with a single thread.

Modules	Time per frame
Feature extraction (GPU implementation)	$\sim$ 30ms
Feature matching with camera pose estimation	$\sim$ 25ms
Keyframes & 3D points updating	$\sim$ 35ms

We have conducted experiments with several challenging examples on a desktop PC with a Xeon E-1230V2 3.3GHZ CPU (4 cores), 8GB memory, and a NVIDIA GeForce GTX560SE display card (1GB memory). The live sequences are captured by a Logitech C905 web camera. Table 3 shows the time spent in different steps with a single thread. With multi-thread programming, the frame rate of our system can be around 25fps.

We first show an indoor example in Figure 1. In this example, a person is rearranging the objects on the desk, and the flashlight irradiate the desk in a back and forth way. It is very challenging for real-time monocular SLAM. We found that PTAM quickly failed in this example, as shown in the bottom row of Figure 1. In contrast, our system can robustly recover the camera poses. Figure 10 shows the reconstructed 3D points and selected keyframes in different time instances. The changed 3D points are recognized and highlighted with red color.

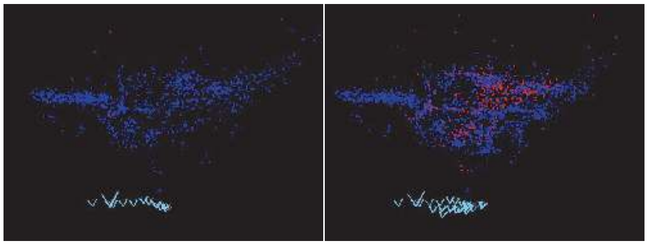

Figure 10: 3D points and keyframes updating. Left: the reconstructed 3D points with selected keyframes when frame 140 is inputted. Right: the reconstructed 3D points with selected keyframes when frame 398 is inputted. The red points are those detected invalid 3D points.

We give a comparison of the same system with/without the component of removing invalid 3D points and keyframes, as shown in Figure 9. As can be seen, without removing invalid 3D points and keyframes, the camera poses of many frames cannot be accurately estimated, so the inserted virtual object (a purple sphere) drifts with the moving books, as shown in Figure 9(b). In contrast, the SLAM result with the component of removing invalid 3D points and keyframes does not have this problem, as shown in Figure 9(a). In this example, since the books are moving quickly and constantly,

[1] The supplementary video can be downloaded from the website http://www.cad.zju.edu.cn/home/gfzhang/

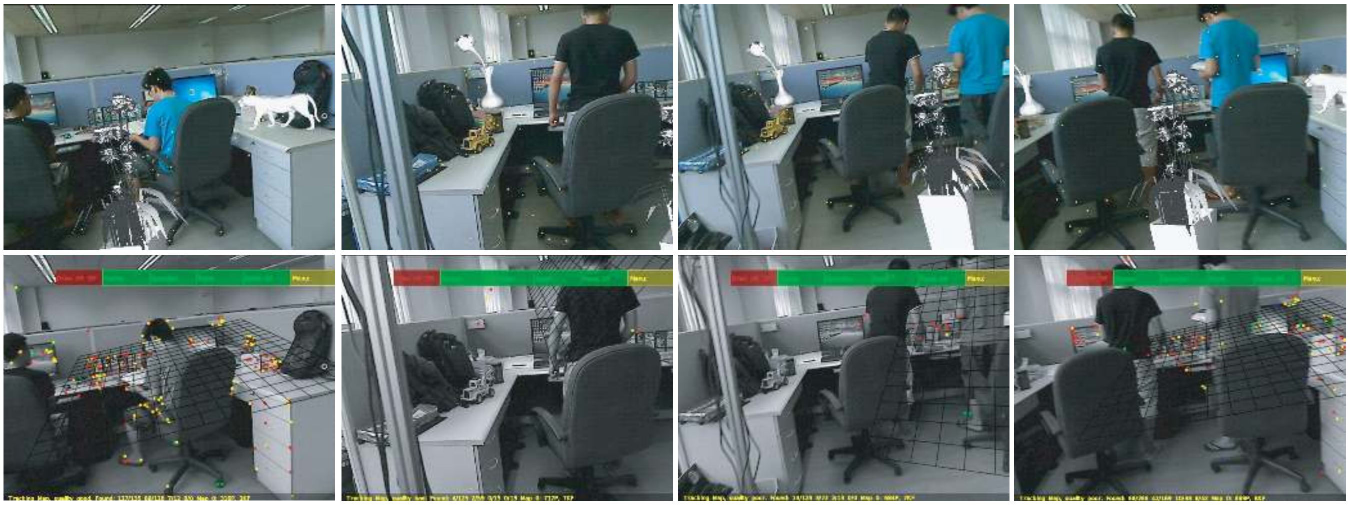

Figure 11: “Two-Men” example. Top: the SLAM result (with augmented reality) of our system. Bottom: the SLAM result of PTAM.

$r_1 = 20$ is not large enough to remove the changed 3D points in time. So we adjust r_1 larger (i.e. $r_1 = 40$) to remove the changed 3D points more reliably. Again, the SLAM result by PTAM has serious drift problems as shown in Figure 9(c).

Figure 11 shows another challenging example. Besides dynamic objects, many regions are quite textureless. Our system also significantly outperforms PTAM in this example. Please refer to our supplementary video for better presentation.

7 Discussions and Conclusions

In this paper, we have presented a novel robust monocular SLAM in dynamic environments. Different to traditional SLAM or PTAM methods, our system employs SIFT features with keyframe representation, which is more reliable for fast camera movement and global relocalization. The changed regions are reliably detected by projecting the nearby keyframes to current frame for appearance and structure comparison. The invalid 3D points and keyframes can be effectively removed and updated in time, so that our system can robustly deal with the scene that is gradually changing. We also propose a prior-based adaptive RANSAC method to efficiently find the inlier matches even the inlier ratio is rather low. Based on the above techniques, our system can work in the challenging situation where there are many large moving objects and significant local illumination variation, which is very difficult for previous works.

Our system still has some limitations. If an object is moving too fast, for the point $\mathbf{x}$ on that object, the default r_1 may be not large enough to detect a neighboring point $\mathbf{y}$, so our system will think $\mathbf{x}$ is occluded by another moving object and keep $V(X) = 1$. Generally, a few changed 3D points in this case does not influence the SLAM result. However, if there are too many such points that have changed but still keep in the map, and the moved objects are still in the scene, our feature matching will have ambiguity and SLAM may fail. An extreme case is that most objects are quickly changing and there are no sufficient static points for robust camera pose estimation. In this case, our system may fail to accurately recover camera poses. In addition, like PTAM, our current system is also limited to work in a small space since real-time bundle adjustment is still intractable for a large-scale scene. In addition, in our current implementation, we build a KD-Tree for all reconstructed 3D features. Since SIFT feature matching by the 2NN heuristic excessively relies on the feature distinctiveness, it may have problems if the number of 3D features in the KD-Tree becomes very large or the scene contains many repeated structures. In our future work, we would like to resolve this problem and further extend our system to support incorporating scene priors (e.g. the reconstructed 3D features in an offline stage like [9]), so that the system may work in a larger space.

Acknowledgements

The authors would like to thank all the reviewers for their constructive comments to improve this paper. This work is partially supported by the 973 program of China (No. 2009CB320804), NSF of China (No. 61103104), the Fundamental Research Funds for the Central Universities (No. 2013FZA5015), and the China Postdoctoral Science Foundation funded project (No. 20110491780).

References

[1] C. Bibby and I. Reid. Simultaneous localisation and mapping in dynamic environments (SLAMIDE) with reversible data association. In *Proceedings of Robotics Science and Systems*, 2007.

[2] G. Bleser, H. Wuest, and D. Stricker. Online camera pose estimation in partially known and dynamic scenes. In *ISMAR*, pages 56–65, 2006.

[3] 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.

[4] D. Chekhlov, M. Pupilli, W. Mayol-Cuevas, and A. Calway. Real-time and robust monocular SLAM using predictive multi-resolution descriptors. In *2nd International Symposium on Visual Computing*, 2006.

[5] T.-J. Chin, J. Yu, and D. Suter. Accelerated hypothesis generation for multistructure data via preference analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(4):625–638, 2012.

[6] O. Chum and J. Matas. Matching with prosac - progressive sample consensus. In *CVPR (1)*, pages 220–226, 2005.

[7] O. Chum, J. Matas, and J. Kittler. Locally optimized ransac. In *DAGM-Symposium*, pages 236–243, 2003.

[8] 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.

[9] Z. Dong, G. Zhang, J. Jia, and H. Bao. Keyframe-based real-time camera tracking. In *ICCV*, pages 1538–1545, 2009.

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

[11] C. Engels, H. Stewénius, and D. Nistér. Bundle adjustment rules. *Photogrammetric computer vision*, 2, 2006.

[12] M. Fiala. ARTag, a fiducial marker system using digital techniques. In *CVPR*, volume 2, pages 590–596, 2005.

[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.

[14] D. Hahnel, R. Triebel, W. Burgard, and S. Thrun. Map building with mobile robots in dynamic environments. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1557–1563, 2003.

[15] E. Imre, J.-Y. Guillemaut, and A. Hilton. Moving camera registration for multiple camera setups in dynamic scenes. In *BMVC*, pages 1–12, 2010.

[16] A. Kawewong, N. Tongprasit, S. Tangruamsub, and O. Hasegawa. Online and incremental appearance-based SLAM in highly dynamic environments. *Int. J. Rob. Res.*, 30(1):33–55, 2011.

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

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

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

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

[21] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fast-SLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *IJCAI*, pages 1151–1156, 2003.

[22] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd. Real time localization and 3D reconstruction. In *CVPR*, volume 1, pages 363–370, 2006.

[23] D. Myatt, P. Torr, S. Nasuto, J. Bishop, and R. Craddock. NAPSAC: High noise, high dimensional robust estimation - it's in the bag. In *BMVC*, pages 458–467, 2002.

[24] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, P. K. Andrew J. Davison, J. Shotton, and S. Hodges. KinectFusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011.

[25] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *ICCV*, 2011.

[26] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(6):756–777, 2004.

[27] K. Pirker, M. R̈uther, and H. Bischof. Histogram of oriented cameras - a new descriptor for visual slam in dynamic environments. In *BMVC*, 2010.

[28] K. Pirker, M. R̈uther, and H. Bischof. CD SLAM - continuous localization and mapping in a dynamic world. In *International Conference on Intelligent Robots and Systems*, pages 3990 – 3997, 2011.

[29] T. Pollard and J. L. Mundy. Change detection in a 3-D world. In *CVPR*, 2007.

[30] M. Pupilli and A. Calway. Real-time camera tracking using known 3D models and a particle filter. In *International Conference on Pattern Recognition*, August 2006.

[31] R. J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam. Image change detection algorithms: A systematic survey. *IEEE Transactions on Image Processing*, 14:294–307, 2005.

[32] E. Rosten, R. Porter, and T. Drummond. FASTER and better: A machine learning approach to corner detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32:105–119, 2010.

[33] Z. Shi, Z. Liu, X. Wu, and W. Xu. Feature selection for reliable data association in visual slam. *Machine Vision and Application*, 24:667–682, 2013.

[34] J. Shimamura, M. Morimoto, and H. Koike. Robust vSLAM for dynamic scenes. In *MVA*, pages 344–347, 2011.

[35] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3D. *ACM Trans. Graph.*, 25(3): 835–846, 2006.

[36] H. Strasdat, J. Montiel, and A. J. Davison. Visual slam: Why filter? *Image and Vision Computing*, 30:65–77, 2012.

[37] A. Taneja, L. Ballan, and M. Pollefeys. Image based detection of geometric changes in urban environments. In *ICCV*, 2011.

[38] P. H. S. Torr and A. Zisserman. Mlesac: a new robust estimator with application to estimating image geometry. *Comput. Vis. Image Underst.*, 78(1):138–156, Apr. 2000.

[39] C. Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). `http://cs.unc.edu/ ccwu/siftgpu`, 2007.

[40] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore bundle adjustment. In *CVPR*, pages 3057–3064, 2011.

[41] G. Zhang, X. Qin, W. Hua, T.-T. Wong, P.-A. Heng, and H. Bao. Robust metric reconstruction from challenging video sequences. In *CVPR*, pages 1–8, 2007.

[42] Z. Zhang, R. Deriche, O. D. Faugeras, and Q.-T. Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artif. Intell.*, 78(1-2):87–119, 1995.

[43] D. Zou and P. Tan. CoSLAM: Collaborative visual SLAM in dynamic environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(2):354–366, 2013.