DT-SLAM: Deferred Triangulation for Robust SLAM

Daniel Herrera C.[†], Kihwan Kim[‡], Juho Kannala[†], Kari Pulli[‡], and Janne Heikkilä[†]

[†]University of Oulu [‡]NVIDIA Research

Abstract

Obtaining a good baseline between different video frames is one of the key elements in vision-based monocular SLAM systems. However, if the video frames contain only a few 2D feature correspondences with a good baseline, or the camera only rotates without sufficient translation in the beginning, tracking and mapping becomes unstable. We introduce a real-time visual SLAM system that incrementally tracks individual 2D features, and estimates camera pose by using matched 2D features, regardless of the length of the baseline. Triangulating 2D features into 3D points is deferred until keyframes with sufficient baseline for the features are available. Our method can also deal with pure rotational motions, and fuse the two types of measurements in a bundle adjustment step. Adaptive criteria for keyframe selection are also introduced for efficient optimization and dealing with multiple maps. We demonstrate that our SLAM system improves camera pose estimates and robustness, even with purely rotational motions.

1. Introduction

Modeling an environment, and tracking the camera motion with respect to the environment, is a key component of many mobile vision and augmented reality (AR) applications, and it has been referred to as SLAM (simultaneous localization and mapping [7, 19, 13, 27]) and TAM (tracking and mapping [14, 20]). We introduce a system that extends previous TAM algorithms, making them more robust, removing restrictions on camera motions, and allowing for online merging of disconnected map regions.

SLAM systems can use different kinds of cameras, including monocular [7], stereo [19], and depth-RGB cameras [13, 27]. Having depth at each frame simplifies initialization and pose estimation, but since depth cameras are not as widely available as traditional cameras, monocular TAM and SLAM remain important research areas. Moreover, the depth cameras are often limited by measurement range, ambient outdoor illumination, and material reflectance [18, 24].

The corresponding author's email is dherrera@ee.oulu.fi.

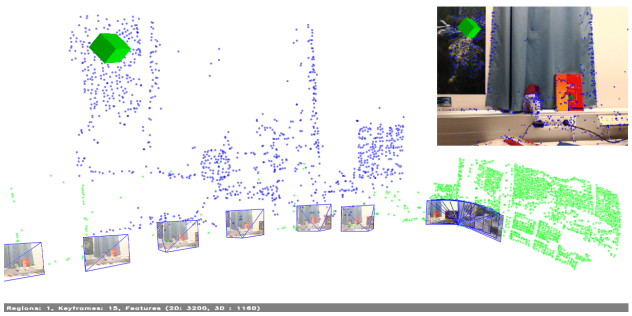

Figure 1. A global view of the structure and estimated camera poses. Triangulated 3D points are shown in blue, 2D features are green (projected onto a unit sphere around the keyframe). A virtual cube is overlaid to simulate an AR application. The features and cube are reprojected on the input video (top-right).

Many TAM systems rely on sparse feature matching and tracking. The point correspondences allow simultaneous estimation of the camera pose and 3D location of the feature points (as in, e.g., PTAM [14]). The availability of source code, its efficiency, and sufficient accuracy have made PTAM popular [6, 2]. Dense SLAM methods [20] require too much computing to be feasible for mobile real-time applications. A client-server solution where much of the computation is shifted to a server [3] could help, but requires high-bandwidth and low-latency connectivity, and uses more energy for wireless data transfer.

Most SLAM systems represent all the model points in 3D. When the baseline between the keyframes is small, the depth ambiguity is high, and the erroneous 3D points corrupt the map and ruin tracking, causing the system to fail. To cope, the systems avoid adding points when the camera is only rotating, but this choice throws away useful information. Another solution is to restart tracking and mapping, but since pure rotations do not constrain the scale of the map, different parts of the map may have different scales.

We provide a new framework that tracks and maps both triangulated (3D) and non-triangulated (2D) features. All tracked features contribute to estimating the pose and building the map. Triangulation of a 2D feature into a 3D point is *deferred* until enough parallax is observed from at least two keyframes. Our key contributions are: **(a)** a unified frame-

work to estimate pose from 2D and 3D features, and incrementally triangulate the points (deferred triangulation), **(b)** a keyframe selection criterion that ensures all new key frames contribute meaningful information, **(c)** a system that handles the inherent problem of undefined scale from optical reconstruction by creating multiple sub-maps and merging them when links are discovered, and **(d)** a real-time scalable implementation released as an open source project.

2. Related Work

Our work concentrates on keyframe-based visual SLAM [26] that works in real time and is suitable for AR applications, as demonstrated by Klein and Murray's PTAM [14] and its extensions [2, 8]. In particular, we do not address completing large structures for navigation [1, 17] or high-quality off-line reconstructions [28]. We focus on improving incremental triangulation with both wide and narrow keyframe baselines, and on better classification of 2D features for efficient sparse scene reconstruction.

Gauglitz *et al*. [10] suggested a visual SLAM system that can handle both parallax-inducing and rotation-only motions. The approach switches between two modes (rotation-only and translation) and creates either 2D panoramas by linking keyframes with homographies, or 3D maps with rigid 3D transformations. However, the tracking module is disconnected from the map: each frame is tracked and registered to the previous frame only. Thus, although the map has triangulated 3D features, only the 2D positions in the previous frames are used by the tracking module. As the authors point out, this compromises tracking robustness, because no global camera tracking or relocalization can be done. An extension [11] uses 3D feature information and estimate an absolute position, but does not use 2D features to improve this absolute position estimate.

Pirchheim *et al*. [22] introduced a Hybrid SLAM system that gives a 3D position to all features, regardless of whether the feature has been triangulated or not. Features that have not been triangulated are assigned an infinite depth. Infinite 3D points correctly constrain only the rotation-component of the pose. However, because the depth is assumed to be known (though infinite), the computed error is incorrectly calculated as the distance between the reprojection and the observation, penalizing parallax between the observations.

As an alternative to classifying a feature as 2D or 3D, it is possible to model the depth uncertainty. This fits well into a filter-based SLAM frameworks because their design naturally includes uncertainty. An inverse depth parametrization produces a measurement equation with improved Gaussianity and thus improves linearity and robustness [5]. However, this parametrization doubles the size of the state vector for each feature, further increasing the already high computational cost of filtering-based methods. The inverse depth parametrization has been used in the context of keyframe-based SLAM to initialize new features [25]. This improves the convergence of feature triangulation. Yet, new features are not used for pose estimation until after initialization. Although our approach could benefit from modeling the depth uncertainty, we focus on using all features for pose estimation without the overhead of modeling the uncertainty.

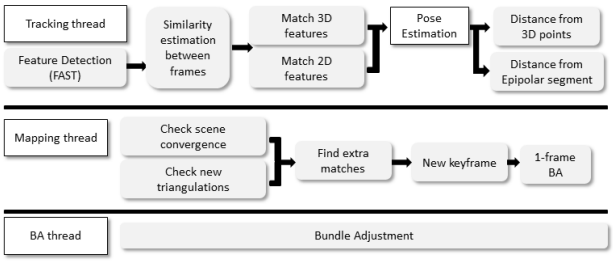

Figure 2. **Overview of our system.**

3. Approach

Our system consists of three parallel threads: tracking, mapping, and bundle adjustment (see Fig. 2).

The tracker attempts to match features in the map to the current image. It assumes that the motion between frames is small so it can use the previous pose estimate to aid in the matching process. The key novelty is the use of both 3D and 2D features in a unified framework. Both are matched by the tracker and both contribute to the pose estimate.

The mapper checks two criteria to determine whether a new keyframe will add meaningful information to the map. It searches for all possible matches from the map. If a 2D feature is observed in the new frame with sufficiently large baseline, it is triangulated. Finally, the mapper refines the pose of the new keyframe and the position of the observed features simultaneously using a 1-frame bundle adjustment.

The bundle adjustment thread constantly runs in the background and simultaneously optimizes all keyframe and feature positions. Bundle adjustment also takes both 2D and 3D features into account using the same unified framework.

To simplify notations we define a function to normalize a vector $\nu(\mathbf{x}) = \mathbf{x}/|\mathbf{x}|$. The notation $\tilde{\mathbf{x}} = [\mathbf{x}, 1]^\top$ augments a vector to homogeneous coordinates and $\tilde{\nu}(\tilde{\mathbf{x}})$ converts back by dividing by the last element and discarding it.

3.1. Camera model

A camera with a very wide field of view can greatly improve the stability of a SLAM system. However, most available webcams have a narrow field of view, and exhibit different types of distortion. We keep our framework independent of the camera model by defining the normalized camera space, a 3D space restricted to the 2D manifold of the unit sphere around the camera. Optical rays (i.e., pixels) can be easily represented as points on this sphere, regardless of how wide the field of view is. Epipolar geometry will be

Figure 3. **Left:** the epipolar segment goes from the projection of the point with minimum depth to that of the point with infinite depth. **Right:** the distance measure has three areas: distance to the line, distance to the minimum depth projection, and distance to the infinite depth projection.

considered in this space because camera distortion causes epipolar lines to become curves in image space.

The camera model defines a function $\phi : \mathbb{R}^3 \to \mathbb{R}^2$ that projects from normalized space to image coordinates (e.g., for the pinhole model $\phi(\mathbf{x}) = \tilde{\nu}(\mathbf{K}\mathbf{x})$). Conversely, $\phi^{-1} : \mathbb{R}^2 \to \mathbb{R}^3$ back-projects onto the unit sphere. Finally, the projection of a 3D point $\mathbf{x}$ to 2D image coordinates $\mathbf{p}$ of a camera with pose $[\mathbf{R}|\mathbf{t}]$ is $\mathbf{p} = \phi(\mathbf{R}\mathbf{x}+\mathbf{t})$.

3.2. The epipolar segment

Our most important contribution is the addition of non-triangulated features to the map. Say a feature is observed in frame 0 at position $\mathbf{m}_0$ and in frame k at position $\mathbf{m}_k$. We wish to utilize all constraints imposed by this match. The traditional formulation of the epipolar equation constrains the matches to lie on a line. Given the relative pose $\mathbf{R}$ and $\mathbf{t}$ between the two frames, we can construct the epipolar line ℓ in normalized space coordinates for frame k:

$$\ell^\top = \nu([\mathbf{t}]_\times \mathbf{R}\,\phi^{-1}(\mathbf{m}_0)). \tag{1}$$

The epipolar lines get warped into complex curves due to camera distortion. Note that in normalized space the epipolar line can be viewed as a plane through the origin with normal ℓ. To avoid distortion, we calculate the displacement vector to the line in normalized space:

$$\mathbf{d}_\ell = (\ell\,\phi^{-1}(\mathbf{m}_k))\ell. \tag{2}$$

We use the first-order Taylor approximation of ϕ to transform this displacement to pixel units. With $\mathbf{J}_{\phi(\mathbf{x})}$ for the 2×3 Jacobian of ϕ evaluated at $\mathbf{x}$, the squared distance to the epipolar line is

$$E_{line} = \|\mathbf{J}_{\phi(\mathbf{m}_k)}\mathbf{d}_\ell\|^2. \tag{3}$$

As Chum *et al*. [4] mention, Eq. (3) does not exploit all the constraints because it permits points behind the cameras. They propose an oriented epipolar constraint that turns the epipolar line into a half-line. We take this notion further and show that a better constraint is an epipolar segment. We then apply this epipolar segment constraint to our matching, pose estimation, and bundle adjustment stages.

A point on an image implicitly defines an optical ray that leaves from the camera center and extends to infinity. Given that we know the relative pose between two cameras, we project any point along the ray on the second image, see Fig. 3. The center of the reference camera (i.e., a point with zero depth) projects to the epipole. There is also a projection on the epipolar line corresponding to infinite depth. These two points define the epipolar segment. Any point behind the epipole or farther than the infinite projection along the epipolar line should be penalized.

We also notice that most of the epipolar segment corresponds to points very close to the camera, especially when the distance between the cameras is small. We can optionally set a minimum depth for all points in the scene $z_{\min}$. This may remove a large part of the epipolar segment, improving matching performance and increasing robustness, with no meaningful impact on the system's flexibility.

When applying this constraint during pose estimation and bundle adjustment, it is important to have a smooth distance measure. Marking points that fail the oriented epipolar constraint, as suggested in [4], makes the system unstable when points are close to the epipole and may bias the system. We propose a smooth distance measure that depends on the distance along the epipolar line, λ:

$$\mathbf{m}_{\min} = \phi(\mathbf{R}(z_{\min}\phi^{-1}(\mathbf{m}_0)) + \mathbf{t}) \tag{4}$$

$$\mathbf{m}_\infty = \phi(\mathbf{R}(\phi^{-1}(\mathbf{m}_0))) \tag{5}$$

$$E_{2D} = \begin{cases} ||\mathbf{m}_{\min} - \mathbf{m}||^2 & \lambda \le \lambda_{\min} \\ ||\mathbf{m}_\infty - \mathbf{m}||^2 & \lambda \ge \lambda_\infty \\ ||J_{\phi(\mathbf{m}_k)}\mathbf{d}_\ell||^2 & \text{else,} \end{cases} \tag{6}$$

where $\mathbf{m}_{\min}$ and $\mathbf{m}_\infty$ are the minimum and the infinite projections. If the match is between the endpoints of the epipolar segment we use the distance to the line, otherwise we use the distance to the closest endpoint, see Fig. 3.

The formulation of Eq. (6) is very important. It allows the observation to drift along the epipolar line without penalizing it and permits the feature to remain an inlier as it smoothly transitions from 2D to 3D and shows more and more parallax. It also penalizes points with negative depth but remains continuous and smooth, which makes it suitable for an iterative minimizer.

3.3. Image-based similarity estimation

We assume the frame rate to be high enough so that the change of pose between frames is not arbitrarily large. Thus, we expect successive images to be fairly similar. When a new frame arrives, the tracker performs an image-based registration like Klein and Murray [15]. However, instead of trying to estimate a 3D rotation, we estimate only a 2D similarity $\mathbf{S}$. This gives us a mapping from the current frame to the previous frame, which we can use to make an initial guess of feature positions in the current frame, given that we know the pose of the previous frame.

3.4. Feature matching

Our matching procedure is similar to Klein and Murray [14], but we extend it to use 2D features for tracking. Matching of a feature consists of two stages: selecting the candidate locations and comparing patches. Each frame has a collection of FAST keypoints [23]. The first stage decides which of these keypoints will be compared to the reference patch. The second stage compares the selected locations and decides which are matches. The comparison stage remains mostly unchanged from [14]. We apply an affine warp to the original feature and compare it to a rectangular patch in the current image using zero-mean sum of squared differences (ZSSD). We perform a sub-pixel refinement of the match position to minimize the ZSSD score s. Denoting the best score by $\hat{s}$, a given candidate is considered a match if $s < \min(\tau_s, \tau_{\hat{s}}\hat{s})$, where τ_s and $\tau_{\hat{s}}$ are constant thresholds.

To select the candidate locations we first transform all keypoint positions $\mathbf{q}_k$ from the current frame into the previous frame's image coordinates using the similarity $\mathbf{S}$ obtained in Sec. 3.3 (i.e., $\mathbf{q}_{k-1} = \mathbf{S}\tilde{\mathbf{q}}_k$). We then calculate the distance from the transformed keypoint to the expected feature position. We do this on the previous frame's image coordinates because we know its pose, and we can project a 3D feature directly from the map to a single position $\mathbf{p}$ on the previous frame. The distance is then $|\mathbf{p} - \mathbf{q}_{k-1}|$. For a 2D feature the distance to the epipolar segment is calculated using Eq. (6). Keypoints with a distance below τ_d are considered as candidate locations for this feature.

To speed up matching, all features that were matched in the previous frame use this matched position as the projected position $\mathbf{p}$ regardless of the feature type. This allows very efficient tracking of matched 2D features.

Note that features can potentially match many locations, especially in the case of 2D features with repetitive texture. Sometimes the rigid-scene constraint and the estimated motion will discard some of these matches as outliers. Occasionally, more than one match position agrees with the camera motion and scene structure. We do not discard or ignore these matches, but keep them and use them throughout the system. For example, each match location that has been marked as an inlier during pose estimation contributes another $\mathbf{p}$ for matching, and candidates are selected from around all the matched positions.

3.5. Choosing a motion model

The system can estimate three types of camera poses: pure rotation, essential matrix, and full 6 DoF pose. The type selected depends on the features matched and motion observed. If enough 3D features are observed, a full pose can be estimated (rotation, translation direction, and scale). If only 2D features are observed and they show enough parallax, an essential matrix can be estimated. This corresponds to a pose with known rotation and translation direction but arbitrary scale. If no parallax is observed in the 2D features, the translation direction cannot be estimated, and we assume a pure rotation with unchanged camera center.

Note that a pure 3D rotation is more restrictive than a homography as used by Gauglitz *et al*. [10], and this is why we can still localize these frames in the current map. Moreover, since the system does not penalize drift of 2D features along the epipolar geometry regardless of the motion model, we are able to smoothly transition from one model to the other. Thus, the model selection is not as crucial as in [10] and we can simply set a threshold on the outlier count to decide between an essential matrix or a pure rotation.

3.6. Pose estimation

Pose estimation begins with a RANSAC stage. The hypothesis generation depends on the motion model. A full pose model uses a 4+1 method based on [9], where 4 matches are used to estimate the pose and 1 is used to verify it. To obtain an essential matrix we use the 5-point algorithm [21] plus 1 match to verify the model. Finally, a pure rotation is easily estimated from 2 points by solving for the absolute orientation of the matches in normalized space.

We refine the pose by minimizing over rotation and translation a cost function built from the observed 2D and 3D features. In case of a pure rotation model, the camera center is fixed. The error for a feature with 3D position $\mathbf{x}$ is the squared distance between its projection and the observed position $\mathbf{m}_k$:

$$E_{3D} = ||\phi([\mathbf{R}_k|\mathbf{t}_k]\tilde{\mathbf{x}}) - \mathbf{m}_k||^2. \tag{7}$$

The error for a 2D feature observed in the image at $\mathbf{m}_k$ depends on the relative pose between the current frame k and the original frame where the 2D feature was created:

$$\mathbf{R}_{0\to k} = \mathbf{R}_k\mathbf{R}_0^\top, \tag{8}$$

$$\mathbf{t}_{0\to k} = t_k - \mathbf{R}_{0\to k}t_0. \tag{9}$$

Using this relative pose we can construct the epipolar plane in normalized space coordinates using Eq. (1), and the distance between its projection and the observed position is calculated using Eq. (6).

The pose is obtained as a minimization of the errors of all matched features

$$\arg\min_{\mathbf{R}_k,\mathbf{t}_k} \sum_i \rho(E_{3D,i}) + \sum_j \rho(E_{2D,j}), \tag{10}$$

where $\rho(c) = \tau_c \log(1 + c/\tau_c)$ is a robust function inspired by the Cauchy distribution that reduces the impact of outliers (measurements with an error larger than τ_c).

Note that for 2D features observed during a pure rotation $\mathbf{m}_{min}$ and $\mathbf{m}_\infty$ project to the same point. Thus Eq. (6) reduces to a single point distance $||\phi(\mathbf{R}_{0\to k}\phi^{-1}(\mathbf{m}_0)) - \mathbf{m}_k||^2$. We detect this and simplify the computation.

3.7. Map regions

As Gauglitz *et al*. [10] point out, the problem of arbitrary scale prevents us from creating one unified global map. However, since our tracker can use information from the current 3D map, there is no need to create "homography groups" as they do. Instead, we recognize that all frames must have a Euclidean pose (rotation and translation) in the map. This is similar to [22], but instead of having groups of panoramas, we only fix the translation component of the frames that do not exhibit enough parallax with any view, there is otherwise no distinction for this frame in the map. Moreover, we are able to detect when a translation of the camera would create scale inconsistencies, i.e., when the camera translates while observing only 2D features. We then create a new separate region so that information can continue to be collected.

The system will try to find new matches between keyframes in the background. If enough matches are found between keyframes of different regions they can be merged. If enough matches are found between unrelated keyframes in the same region, a loop is detected.

3.8. Adding new keyframes

The system evaluates incoming frames in the background to determine whether they should be added as a keyframe. Because the map also contains non-triangulated features, we now have much better criteria for adding a keyframe. By matching the 2D features we are able to determine which areas of the new image are already covered by features in the map. When a new frame is being considered as a keyframe, we attempt to match features from the map until one of the following conditions happens or there are no more features to match:

- A given number of new 2D features (τ_n) can be created from areas not covered by the map.
- A given number of 2D features (τ_t) can be triangulated.
- A given number of 3D features (τ_t) have been observed from a significantly different angle.

If any of these criteria is fulfilled, we add the frame to the map. This ensures that a new keyframe will always contribute meaningful information to the map and reduces unnecessary and redundant information.

3.9. Deferred triangulation of 2D features to 3D

When a new 2D feature is added to the map, the position $\mathbf{p}$ on the source image is stored, which along with the camera pose $\mathbf{T}$ describes the optical ray along which the feature must be located in 3D. We defer triangulating a 2D feature until we observe another keyframe with sufficient parallax. When a new keyframe containing a measurement of the feature is added to the map, the angle between the optical rays is evaluated. If the angle is above a given threshold τ_α, the feature is considered to have enough baseline to be accurately triangulated. This allows the system to flexibly determine which points should be triangulated and which not. The 3D position $\mathbf{x}$ is calculated as the closest point to both optical rays, and the feature becomes a 3D feature.

3.10. Bundle adjustment

Bundle adjustment of the entire map runs in the background. It minimizes a cost similar to Eq. (10), but over all camera poses and point positions (with $\mathscr{R} = \{\mathbf{R}_0 ... \mathbf{R}_K\}$, $\mathscr{T} = \{\mathbf{t}_0 ... \mathbf{t}_K\}$, and $\mathscr{X} = \{\mathbf{x}_0 ... \mathbf{x}_N\}$):

$$\underset{\mathscr{R},\mathscr{T},\mathscr{X}}{\arg\min} \sum_{k \to K} \left(\sum_{i \to M} \rho(E_{3D,k,i}) + \sum_{j \to N} \rho(E_{2D,k,j}) \right). \quad (11)$$

Each sub-map will be adjusted independently. Frames that observed only 2D features with no parallax have their camera centers fixed to that of the reference keyframe.

4. Results and Evaluation

We provide a quantitative evaluation by testing our system with videos from the *City of Sights* [12], a digital urban scene developed for AR research. This dataset provides ground truth camera trajectories captured by moving a camera using a robot arm. We also provide a qualitative comparison to previous work by testing with the raw video from Pirchheim *et al*. [22] and our own captured scenes.

Table 1 shows the average timings for the different components of our system. We test the system on a Core i7 running at 3.5 GHz. The critical component is clearly feature matching. We note that our code has not been particularly optimized. PTAM tracks 1000 features in real time, but its code includes hand-crafted SSE assembly optimizations for the matching code. Yet, even with only 400 features we show improved accuracy in the following results. Similar optimizations as in [16] can be used to run our system on a mobile platform.

Figure 4 shows a comparison of ground truth camera trajectory against the estimated camera poses from our system and PTAM. In each graph, insets show that the estimated pose from our system is more accurate and stable than PTAM, achieving consistently a lower RMSE.

Figures 5 and 6 show additional qualitative results that represent how our approach can handle pure rotation even without stereo initialization, and how we merge the tracks of 2D features in a 3D structure, which is not possible in PTAM [14] or Hybrid SLAM [22]. The examples also demonstrate how our system can handle different submaps from the same scene, indicating the capability of loop closure and efficient relocalization.

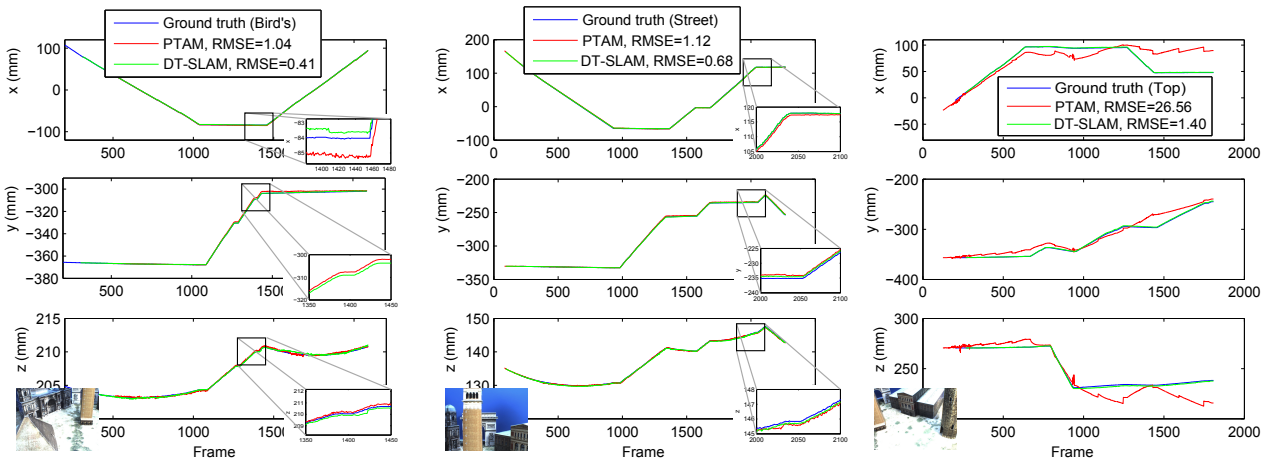

Figure 4. **Estimated camera pose.** (Scenes from the City of Sights [12]). **Left:** Bird's eye scene. **Center:** Street view scene. **Right:** Top view scene. Each graph shows the camera trajectory. An inset image in the middle shows the difference between PTAM and our approach. Note that our estimated pose is closer to the ground truth and shows less drift.

Table 1. **Performance numbers for DT-SLAM.** Left duration tracks 400 features, right duration tracks 200 features. Bundle adjustment used 30 keyframes and 2300 features.

Thread	Component	Duration (ms)	
Tracking	Similarity	0.17	
	Feature detection	2.66	
	Feature matching	22.20	12.03
	Pose RANSAC	4.31	3.38
	Pose refinement	8.48	6.18
	Total	**37.82**	**24.42**
New keyframe	Feature matching	41.62	
	Pose refinement	31.8	
	1-frame BA	193.00	
	Total	**265.70**	
Bundle adjustment		**666.07**	

Finally, a qualitative comparison with Pirchheim *et al.* [22] is shown in Fig. 7. The sequence contains many pure rotations. PTAM gets lost at each rotation. Hybrid SLAM keeps tracking but is unable to use the observed features during rotation for triangulation. Our system can triangulate more features by matching features from different rotations. More results can be found in the supplemental video.

5. Discussion

We have shown that our approach provides more accurate and stable results than PTAM, one of the most popular tracking and mapping methods, by quantitatively and qualitatively evaluating various datasets. Because the source code for the most relevant previous works [10, 22] is not available, we are unable to provide a quantitative comparison with them. However, by comparing with the results shown in Pirchheim *et al.*'s paper (Fig. 7) we can qualitatively show how our approach improves upon theirs.

The results from that video highlight two key differences between our system and Hybrid SLAM. First, as Pirchheim *et al.* mention, their panorama estimates are not first class citizens of the SLAM map. This means that they cannot use the information from two panorama maps to triangulate features, whereas our system does, resulting in a more complete map and a more accurate pose estimate. Second, they force 2D features to have infinite depth which penalizes stereo displacement. This results in noticeable jitter when the system assumes a pure rotation and the camera translates. Our system demonstrated the ability to cope with this translation by using a flexible 2D error measure (Eq. (6)) and multiple regions.

We consider our system as the logical next step from the contributions of Gauglitz *et al.* [10] and Pirchheim *et al.* [22]. Those approaches switch the pipeline between 6DOF tracking and panoramic modes [10] or force 2D features into 3D [22], whereas our approach generalizes the use of both rotation-only (3DOF) and general 6DOF camera motion into a unified framework. Our mapping module combines the idea of avoiding relocalization by keeping multiple regions and merging them, but is also able to fuse the information from different 3D rotations into a single global coordinate frame. Our tracking module takes advantage of the optimized map to establish 2D-3D matches and robustly estimate a 6 DoF pose, yet we do not penalize stereo displacement for non-triangulated features. Our system is more robust to errors in the motion model selection because we can smoothly transition between models while tracking pure rotations in the same Euclidean space. We also provide more effective keyframe selection criteria than conventional keyframe-based SLAM, which often add redundant keyframes. This directly affects the amount of computation needed for optimization stages. Finally, our

bundle adjustment component takes into account all observations from both 2D and 3D features to obtain the best reconstruction possible.

However, because no implementations of [10, 22] are available, we were only able to directly compare the performance against PTAM. This has also motivated us into making the source code of our system available to other researchers, both to help them getting started in building a SLAM system, and for comparing their system against ours.

It is also worth noting a few of limitations in our work. Because we rely on keypoint-based visual feature tracking, inherently, our approach would not work properly in textureless scenes. If a depth camera were available it could be integrated into our tracking and mapping framework. Secondly, the matching stage is the weakest link of the system. It consumes most of the computation time and is not as robust as modern feature descriptor matching. Finally, drastic rotations may violate our assumption of smooth motion. Integrating inertial sensors to give an initial guess for the camera pose would increase robustness.

6. Conclusion

We introduced a new keyframe-based visual SLAM system which handles camera motions from both pure rotation and translation. Even with feature correspondences separated only by a narrow baseline, our system tracks the features locally and incrementally, and triangulates the features once they are observed in a new keyframe with sufficient baseline. Therefore, the proposed system does not require an explicit stereo initialization, and gradually converges to stable tracking and pose estimation. The evaluation shows that our approach provides a more robust and stable pose estimation than previously reported keyframe-based SLAM systems. We hope that many computer vision applications that need efficient camera pose estimation will benefit from availability of the source code.

References

[1] T. Bailey and H. Durrant-Whyte. Simultaneous localisation and mapping (SLAM): State of the art. In *Robotics and Automation Magazine*, 2006.

[2] R. O. Castle, G. Klein, and D. W. Murray. Wide-area augmented reality using camera tracking and mapping in multiple regions. *CVIU*, 115(6), 2011.

[3] J. Chen, D. Bautembach, and S. Izadi. Scalable real-time volumetric surface reconstruction. *ACM TOG*, 32(4), 2013.

[4] O. Chum, T. Werner, and J. Matas. Epipolar geometry estimation via ransac benefits from the oriented epipolar constraint. In *ICPR*, 2004.

[5] J. Civera, A. Davison, and J. Montiel. Inverse depth to depth conversion for monocular slam. *ICRA*, 2007.

[6] A. Davis, M. Levoy, and F. Durand. Unstructured light fields. *Eurographics*, 2012.

[7] A. Davison, I. Reid, N. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE PAMI*, 29(6), 2007.

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

[9] X. Gao, X. Hou, J. Tang, and H. Cheng. Complete solution classification for the perspective-three-point problem. *IEEE PAMI*, 2003.

[10] S. Gauglitz, C. Sweeney, J. Ventura, M. Turk, and T. Höllerer. Live tracking and mapping from both general and rotation-only camera motion. *ISMAR*, 2012.

[11] S. Gauglitz, C. Sweeney, J. Ventura, M. Turk, and T. Höllerer. Model estimation and selection towards unconstrained real-time tracking and mapping. *TVCG*, 20, 2014.

[12] L. Gruber, S. Gauglitz, J. Ventura, S. Zollmann, M. Huber, M. Schlegel, G. Klinker, D. Schmalstieg, and T. Höllerer. The city of sights: Design, construction, and measurement of an augmented reality stage set. In *ISMAR*, 2010.

[13] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *UIST*, 2011.

[14] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *ISMAR*, 2007.

[15] G. Klein and D. Murray. Improving the agility of keyframe-based slam. In *ECCV*, 2008.

[16] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. In *ISMAR*, 2009.

[17] J. Kwon and K. M. Lee. Monocular SLAM with locally planar landmarks via geometric Rao-Blackwellized particle filtering on Lie groups. In *CVPR*, 2012.

[18] B. Langmann, K. Hartmann, and O. Loffeld. Depth camera technology comparison and performance evaluation. In *ICPRAM*, 2012.

[19] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid. A constant time efficient stereo slam system. In *BMVC*, 2009.

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

[21] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE PAMI*, 26(6), 2004.

[22] C. Pirchheim, D. Schmalstieg, and G. Reitmayr. Handling pure camera rotation in keyframe-based slam. *ISMAR*, 2013.

[23] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *ICCV*, 2005.

[24] J. Shen and S.-C. S. Cheung. Layer depth denoising and completion for structured-light rgb-d cameras. In *CVPR*, 2013.

[25] H. Strasdat, J. Montiel, and A. Davison. Scale drift-aware large scale monocular slam. *Robotics: Science and Systems*, 2010.

[26] H. Strasdat, J. M. M. Montiel, and A. Davison. Real-time monocular SLAM: Why filter? In *ICRA*, 2010.

[27] Y. Taguchi, Y.-D. Jian, S. Ramalingam, and C. Feng. SLAM using both points and planes for hand-held 3d sensors. In *ISMAR*, 2012.

[28] H. H. Vu, P. Labatut, J.-P. Pons, and R. Keriven. High accuracy and visibility-consistent dense multi-view stereo. *IEEE PAMI*, 2012.

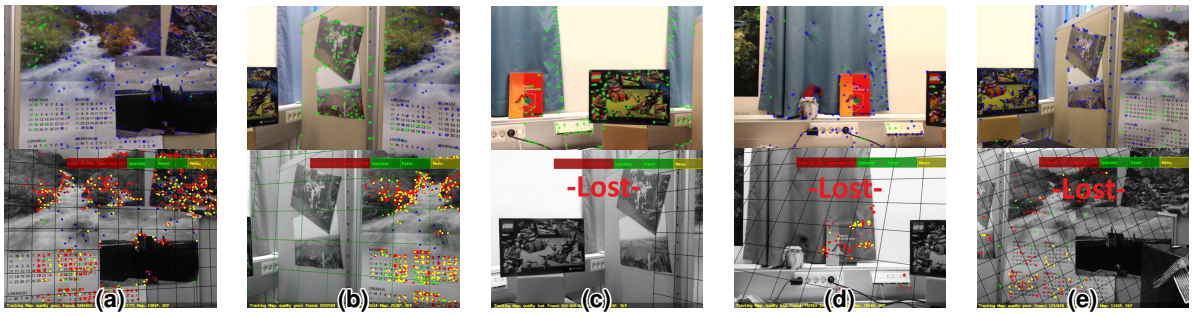

Figure 5. Normal workflow with translational and rotational motion. **Top row:** DT-SLAM. **Bottom row:** PTAM. **(a)** Tracking with translational motion. **(b)** Pure rotation begins. No new features can be triangulated. **(c)** PTAM gets lost due to the lack of triangulated features. Our system estimates a rotation relative to the previous key frame. **(d)** A new region is created when enough parallax is observed. PTAM is still lost. **(e)** Deferred 2D features are triangulated (blue points in the middle). When the two regions overlap they are merged into one (See Fig. 6). PTAM attempts to relocalize but fails.

Figure 6. **Region merging after a pure rotation.** Blue represents 3D features and frames from the first region, purple are from the second region. Green dots are non-triangulated features (projected on a unit sphere around the keyframe for display only). **Left:** First region (keyframes and features) from the starting frame of sequences shown in Fig. 5 to the frame where camera only rotates (Fig. 5(b)(c); denoted as a dotted red line). **Middle:** Second region created after pure rotation. Notice the scale difference between the regions. **Right:** After region merge all keyframes and features are in the same Euclidean space with the same scale. Only when an overlapping region has been triangulated, can both regions be merged into a single coordinate system.

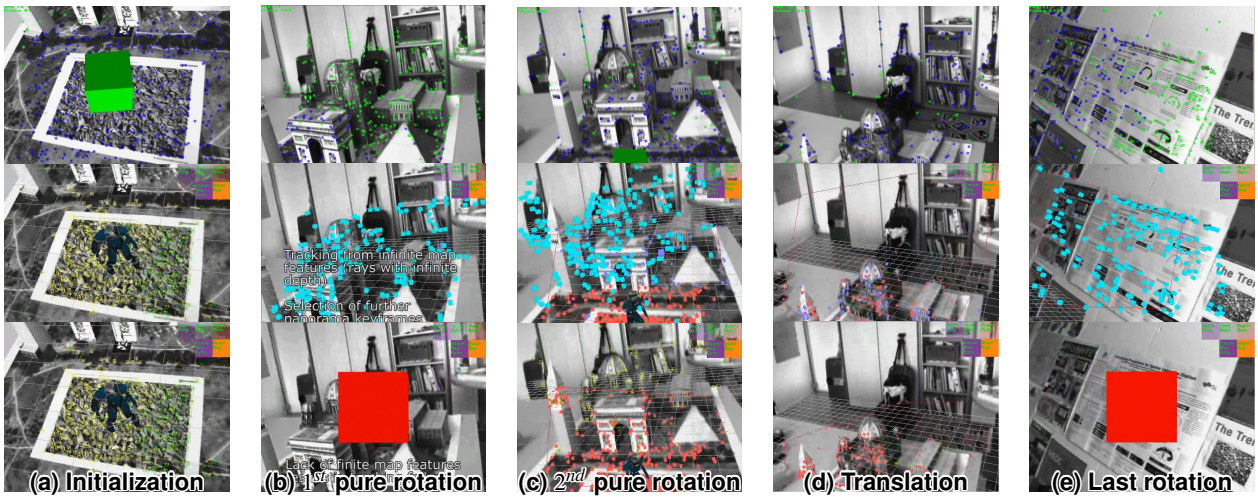

Figure 7. **Qualitative comparison** between DT-SLAM (**Top row**), Hybrid SLAM [22] (**Middle row**), and PTAM [14] (**Bottom row**). We test our approach with the raw video used in [22]. In the top row, triangulated 3D points are blue and 2D features are green, while in the middle and bottom rows they are red and cyan, respectively. **(a)** After stereo initialization, most points are triangulated with a modest amount of translation. **(b)** Pure rotation begins. Points close to the camera have been triangulated in our approach. PTAM gets lost due to the lack of triangulated features. **(c)** When a second rotation motion observes the same area, our system is able to triangulate points observed in the previous rotation. Hybrid SLAM tracks the rotation but cannot combine 2D features and does not triangulate them. Features are forced to have infinite depth, resulting in a lot of jitter (see the supplemental video). PTAM simply relocalized and only tracks the previously triangulated features. **(d)** The camera moves in a regular motion (rotation and translation). All systems work, ours has the most model points. **(e)** Again our system is able to combine two pure rotations to incrementally triangulate features. Hybrid SLAM switched to pure rotation mode, while PTAM is lost again.