Implementing ORB SLAM for Navigation in Indoor Environments

Abizer Patanwala
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester, MA, USA
apatanwala@wpi.edu

Joy Mehta
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester, MA, USA
jjmehta1@wpi.edu

Kunal Nandanwar
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester, MA, USA
kgnandanwar@wpi.edu

***Abstract*—Visual simultaneous localization and mapping (vSLAM) is the technique of concurrently mapping the environment and determining the location and orientation of a camera in relation to its surroundings. The technique merely makes use of the camera's visual inputs. Augmented reality, robotics, and autonomous driving are some uses for vSLAM.**

This study shows how to handle visual data from a monocular camera to create a map of an indoor area and estimate the trajectory of the camera. We employ ORB-SLAM, the feature-based vSLAM algorithm.

***Index Terms*—ORB Slam, SLAM, Mapping**

I. Introduction

Autonomous interior navigation of robots has emerged as one of the major challenges for service-oriented robots as a result of the explosion in industrial automation. Several approaches have been put up to address the issue of interior navigation, including the use of odometers [1], which are prone to drift and lack the loop closing characteristic. Other approaches have suggested using unreliable, expensive-to-integrate devices like lidars [2] and depth sensors [3].

The mapping, localization, path planning, and dynamic obstacle avoidance are components of the visual SLAM problem. There are two ways to solve the Visual SLAM problem: the filtering-based method, which simultaneously modifies the pose and posture of the Key Feature points on the map using techniques like the Extended Kalman Filter [4], Unscented Kalman Filter [5], and Particle Filter [6], FAST SLAM method proposed by Michael Montemerlo et al. [7] as well as Monocular SLAM method proposed by A.J. Davison [8]. The alternative method entails ego-body geometry optimization using error minimization methods, such as PTAM (Parallel Tracking & Mapping) [9] proposed by Georg Klein et al. and, ORB-SLAM [10] and ORB-SLAM2 proposed by Ral Mur-Artal [11], and graph optimization using the bundle adjustment method to map the feature point.

In this paper, we present how to process image data from a monocular camera to build a map of an indoor environment and estimate the trajectory of the camera. We use ORB-SLAM, which is a feature-based Visual SLAM algorithm. We use helper functions from OpenCV to implement ORB SLAM. We choose this algorithm because it is one of the best Visual SLAM algorithms which outperforms many visual inertial algorithms in terms of accuracy as shown in [21]. For benchmarking, the pedestrian dataset is used in [21]. [22] compares various SLAM algorithms for indoor navigation and ORB SLAM again turns out to be one of the best methods for navigation. The high performance of ORB-SLAM in both outdoor and indoor environments makes it a suitable choice to deploy in navigation. Section II presents a literature review and Section III describes the Problem statement.

II. Literature Review

A. Place Recognition

Williams et al. study's [12] evaluated a number of methods for location recognition and came to the conclusion that strategies focused on appearances, such as image-to-image matching, scale better in big contexts than map-to-map or image-to-map approaches. Bags of word strategies [13], such as the probabilistic methodology FAB-MAP [14], are at the forefront of appearance-based methods due to their great effectiveness. For the first time, DBoW2 [15] combined the highly effective FAST feature detector with bags of binary words derived from BRIEF descriptors. Comparing this to the SURF and SIFT features that were previously employed in bags of words techniques, the time required for feature extraction was lowered by a factor of more than one order of magnitude.

B. Map Initialization

Because depth cannot be determined from a single image, monocular SLAM requires a process to produce an initial map. Tracking a well-known structure first can help find a solution to the issue. Using an inverse depth parameterization, points can be initialized with significant depth uncertainty in the context of filtering procedures, with the goal that they will later converge to their actual placements.

Initialization techniques based on two views either assume local planar scenes and determine the relative camera pose from a homography using the Faugeras and Lustman method, or compute an essential matrix that models planar and general scenes using the five-point algorithm of Nister's, which necessitates dealing with multiple solutions. If all points in a planar image are closer to one of the camera centres, both reconstruction approaches suffer from a two-fold ambiguity solution and are not adequately controlled under low parallax. On the other hand, a unique basic matrix can be generated using the eight-point approach if a nonplanar scene is perceived with parallax, allowing the relative camera pose to be accurately determined.

C. Monocular Simultaneous Localization and Mapping

Over the past decades, several prominent monocular SLAM techniques have been proposed such as MonoSLAM [8], PTAM [9], CD-SLAM [19], ORB-SLAM [10], ORB-SLAM2 [11], Edge-SLAM [23], DTAM [24], LSD-SLAM [25] and DSO [26]. Among these only MonoSLAM and Monocular FastSLAM are filter-based techniques. Filtering was initially used to solve monocular SLAM [16]. In that method, the filter processes each frame in order to estimate both the camera pose and the locations of map features. It has the disadvantages of accumulating linearization errors and wasting computing processing consecutive frames with little new information. However, because mapping is independent of frame rate, keyframe-based techniques estimate the map using just selected frames (keyframes), enabling more expensive but accurate BA improvements. Keyframe-based approaches are more accurate than filtering for the same computing cost, according to research by Strasdat et al. [17]. MonoSLAM was one of the first algorithms to implement visual SLAM. It consists of system initialization, followed by an update of the state vector considering the constant velocity model. The camera motion and map of the environment are computed in real-time using the Extended Kalman filter. Majority of the state of the art SLAM techniques are based on visual SLAM which employs optimization techniques rather than filter based. PTAM was a pioneering algorithm in the sense that it was the first to separate tracking and mapping threads and apply keyframes to mapping threads. The mapping thread initialises the map. Then using two consecutive keyframes depth points are estimated using triangulation. The tracking thread calculates camera poses. For each new keyframe, it computes the initial camera pose by minimising the reprojection error of 3D world points projected on the current frame using established correspondences. DTAM was one of the first methods to perform direct tracking and mapping. By defining data cost volume as the average photometric error of several frames computed for the inverse depth of the current frame, the first stage attempts to estimate the depth values. The inverse depth that minimized the photometric error is used for reconstruction. The second stage is responsible for motion computation. It does so by aligning the current frame with the reprojected image from the dense model. LSD-SLAM and DSO also perform direct tracking and mapping.

With a front-end based on optical flow implemented on a GPU, followed by FAST feature matching and motion-only BA, and a back-end based on sliding-window BA, Strasdat et al. [18] presented a large-scale monocular SLAM system. The scale drift that appeared in monocular SLAM was corrected by using a pose graph optimization with similarity constraints [7 degrees of freedom (DoF)] to solve loop closures.

Strasdat et al. [18]' s usage of the PTAM front-end limited the tracking to a local map that was obtained from a covisibility graph. They suggested a back-end for an optimization system with two windows, one for posture graph and the other for continuous BA. However, loop closing only works if the outer window is sizable enough to encompass the entire loop. In our system, we utilize the great concepts of creating a pose graph from the covisibility graph and using a local map based on covisibility, but we do it in a front-end and back-end that have been completely revamped.

Pirker et al. [19] 's proposal for CD-SLAM is a fairly full system with efforts to work in dynamic situations, loop closing, relocalization, and large-scale operation. Initializing the map is not specified, though. We are unable to do a comparison of accuracy, robustness, or large-scale capabilities because there isn't a public implementation.

Song et al. [20]'s visual odometry makes use of a temporal sliding window BA back-end and ORB characteristics for tracking. In order to prevent monocular scale drift, they also use the known distance between the camera and the ground. As opposed to their method, which lacks global relocalization, loop closing, and map reuse, ORB SLAM is more versatile.

ORB-SLAM is an optimization-based method which uses ORB features for mapping, tracking, loop closing and relocalization. It's an extension of PTAM. It performs real-time loop closure based on pose graph optimization. It also uses the survival of the fittest approach for map points and keyframes. This policy improves tracking robustness and enhances life-long operation because redundant keyframes are dropped.

III. Problem Description

We have an indoor static environment, which can be anything such as a home, office, restaurant, warehouse or inside factory. Our aim is to localize a mobile robot and create a map of the surroundings to be used further for motion planning tasks. One particular challenge we face is the lack of textures in the indoor environment. Another challenge which we face is motion planning using a sparse map. We choose to implement ORB-SLAM because of its comparatively high accuracy as compared to many state-of-the-art algorithms and its tolerance for long-hour operations. The pipeline of ORB-SLAM is shown in Fig.1

IV. System Overview

A. Feature Choice

One of the key concepts in our system's design is the use of the same characteristics for location identification, frame-rate relocalization, and loop detection that are utilized for mapping and tracking. By doing so, we may effectively use our method without having to extrapolate the depth of the recognition features from nearby SLAM data.

We selected ORB, which are multi-scale FAST corners that are orientated and have a 256-bit descriptor attached. They offer strong perspective invariance and are very quick to compute and match. This makes it possible to match them across a wide range of baselines, improving BA accuracy.

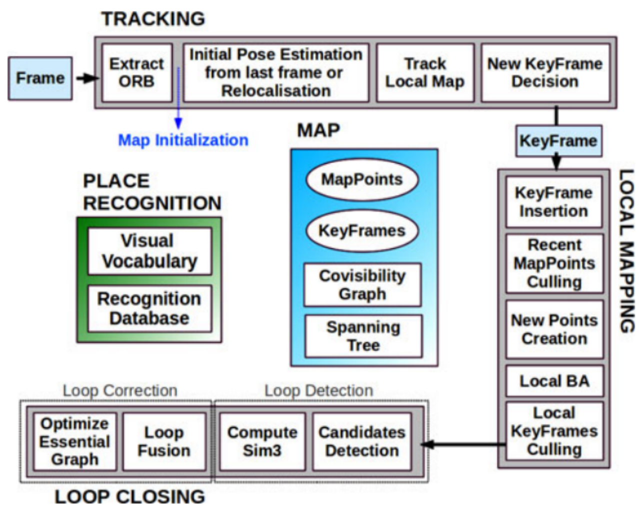

Fig. 1: ORB-SLAM Pipeline showing tracking, mapping and loop closure steps. Components of the place recognition module are also shown. [10]

B. Three Threads: Tracking, Local Mapping and Loop Closing

Our system uses three concurrent threads for tracking, local mapping, and loop closure. For every frame, the tracking is in charge of localizing the camera and choosing when to add a new keyframe. Prior to employing motion-only BA to optimize the posture, we first do an initial feature matching with the preceding frame. The location recognition module is used to carry out global relocalization whenever the tracking is lost (for instance, as a result of occlusions or sudden movements). The system-maintained covisibility network of keyframes is used to extract a local viewable map after an initial assessment of the camera posture and feature matching. Following a reprojection search for matches with the nearby map locations, the camera posture is once more optimized. The tracking thread ultimately selects whether to insert a new keyframe.

New keyframes are processed by the local mapping, and local BA is used to generate the best reconstruction possible around the camera posture. In order to triangulate new points, related keyframes in the covisibility graph are searched for new correspondences for mismatched ORB in the new keyframe. Following creation, a strict point culling strategy is used to ensure that only high-quality points are kept, depending on the data acquired during tracking. Duplicate keyframes must be removed using local mapping.

Every new keyframe triggers a search for loops by the loop closure. When a loop is found, a similarity transformation that describes the drift collected in the loop is computed. The duplicated points are then fused once both sides of the loop have been aligned. To establish global consistency, a pose graph optimization over similarity constraints is then carried out. The primary innovation is that the optimization is carried out over the Essential Graph, a sparser subgraph of the Covisibility Graph.

To do all optimizations, we employ the Levenberg-Marquardt method.

C. Map Points, KeyFrames and their Selection

Each map point p_i stores:

- Its 3D coordinate system location is $X_{w,i}$.
- The direction of observation n_i is the average unit vector of all of its directions of observation (the rays that join the point with the optical centre of the keyframes that observe it).
- A typical ORB description D_i is the related ORB descriptor in the keyframes when the point is detected, and it has the smallest hamming distance compared to the other associated descriptors.
- The scale invariance limits of the ORB features, specify the maximum d_{max} and lowest dd_{min} distances at which the point may be seen.

Each keyframe K_i stores:

- The rigid body transformation known as the camera posture Tiw transfers points from the outside world to the camera coordinate system.
- The internal components of the camera, such as the focal length and main point.
- If a distortion model is given, all ORB features taken from the frame, whether or whether they are linked to a map point, have coordinates that are undistorted.

The creation of map points and keyframes follows a liberal policy, while the detection of superfluous keyframes and incorrectly matched or untrackable map points is handled by a more stringent culling method. This enables a variable map extension during exploration, enhancing tracking resilience under challenging situations (such as rotations and rapid movements), while limiting the size of the map for repeated visits to the same area, or lifetime operation. In addition, despite having fewer points than PTAM, our maps have a relatively low number of outliers.

D. Covisibility Graph and Essential Graph

Our system's covisibility information, which is shown as an undirected weighted graph, is quite helpful for a number of jobs. Each node is a keyframe, and if two keyframes share observations of the same map points (at least 15), then there is an edge between them. The weight θ of the edge is determined by the number of shared map points.

We carry out a pose graph optimization that disperses the loop-closing mistake over the network in order to repair a loop. We suggest creating an Essential Graph, which has all the nodes (keyframes), but has fewer edges, in order to avoid including all of the covisibility graph's potentially dense edges. This would still maintain a robust network that produces correct results. The approach creates a linked subgraph of the covisibility graph with the fewest possible edges by building a spanning tree gradually from the starting keyframe. When a keyframe is added, it is connected to the keyframe that has the most point observations in common. Likewise, when a keyframe is removed due to a culling policy, the system updates the links that were impacted by that keyframe. A robust network of cameras is created by the Essential Graph, which includes the spanning tree, the subset of covisibility graph edges with high covisibility ($\theta_{min} = 100$), and the loop closure edges. The answer is so precise after the pose graph optimization that adding a full bundle adjustment optimization scarcely enhances it.

E. Bags of Words Place Recognition

To conduct loop identification and relocalization, the system incorporates an inbuilt bags of words place recognition module. The descriptor space sometimes referred to as the visual language, may be discretized into visual words. The ORB descriptors that are taken from a sizable collection of photos are used to generate the vocabulary offline. As demonstrated in our prior work, provided the pictures are sufficiently generic, the same language may be utilized for multiple contexts and yet produce acceptable results. The method gradually creates a database with an inverted index, which records each visual word in the lexicon in which keyframes it has been observed. This allows for extremely efficient database searching. When a keyframe is eliminated through the culling process, the database is also updated.

When searching the database, there won't be a distinct keyframe with a high score since keyframes visually overlap. This overlapping was taken into consideration in the original DBoW2 by adding up the score of photos that are near in time. The drawback of this is that keyframes that view the same location but are added at a different time are not included. The keyframes that are related in the covisibility graph are instead grouped. Additionally, any keyframe matches with scores greater than 75% of the top score are returned from our database.

The bags of words model for feature matching was claimed to have further advantages in [5]. When calculating the correspondences between two sets of ORB features, we may speed up the search by limiting the brute force matching to features that are associated with the same node in the vocabulary tree at a certain level (we choose the second out of six). We employ this technique for loop identification and relocalization as well as while looking for matches for triangulating new points. In order to ensure that all correspondences rotate coherently, we further refine the correspondences using an orientation consistency test that eliminates outliers.

V. Automatic Map Initialization

To triangulate a starting set of map points, the map initialization computes the relative posture between two frames. This technique should choose a decent two-view configuration, or a configuration with noticeable parallax, without requiring human interaction and should be independent of the scene (planar or generic). We suggest computing two geometrical models simultaneously: a basic matrix for a non-planar scene and a homography for a planar scenario. We then choose a model using a heuristic and attempt to recover the relative posture for that model using a particular approach. By spotting low-parallax scenarios and the well-known twofold planar ambiguity, our technique avoids initializing a damaged map and only initializes when it is guaranteed that the two-view setup is secure. Our algorithm's stages are as follows:

1) Find initial correspondences
2) Parallel computation of the two models
3) Model Selection
4) Motion and Structure from Motion recovery
5) Bundle adjustment

VI. Tracking

In this part, we go over the actions that the tracking thread takes with each frame it receives from the camera. The motion-only BA makes up the camera pose optimizations, which are discussed in multiple phases.

A. ORB Extraction

At 8 scale levels and a scale factor of 1.2, we extract FAST corners. We discovered that extracting 1000 corners from images with dimensions between 512 x 384 to 752 x 480 pixels was appropriate. The next step is to find corners in each cell, and if not enough corners are identified, we adjust the detector threshold. If certain cells don't have corners, the number of corners kept per cell is likewise modified (textureless or low contrast). Then, using the FAST corners that were kept, the orientation and ORB description are calculated. In contrast to PTAM's search via patch correlation, every feature matching uses the ORB descriptor.

B. Initial Pose Estimation from Previous Frame

If tracking was successful for the previous frame, we forecast the camera posture using a constant velocity motion model and do a guided search of the map points seen there. We utilize a broader search of the map points around their location in the previous frame if not enough matches were discovered (i.e., the motion model is obviously broken). The identified correspondences are then used to optimize the stance.

C. Initial Pose Estimation via Global Relocalization

In the event that tracking is lost, the frame is converted to a bag of words, and keyframe candidates for global relocalization are sought by searching the recognition database. In each keyframe, we compute correspondences with ORB related to map points. Then, for each keyframe, we alternately run RANSAC iterations before attempting to determine a camera posture with the PnP method. We optimize the posture and do a guided search for more matches using the map points of the candidate keyframe if we discover a camera pose with sufficient inliers. Finally, the camera posture is once again adjusted, and the tracking process continues if there are sufficient inliers.

D. Track Local Map

We can project the map into the frame and look for further map point correspondences after we have an estimate of the camera posture and the first set of feature matches. We merely project a local map in order to limit the complexity of huge maps. This local map includes a set of keyframes K_1, which have map points in common with the current frame, as well as a set K_2, which are keyframes K_1's covisibility graph neighbours. Additionally, the local map contains a reference keyframe $K_{ref} \in K_1$ that shares the majority of the map's points with the current frame. Now, every map point visible in K_1 and K_2 is searched as follows in the current frame:

1) Calculate the map point projection **x** for this frame. If it lies outside the image limits, throw it away.
2) Determine the angle between the current viewing ray **v** and the average viewing direction **n** of the map point. If **v . n** $> cos(60)$, discard.
3) Calculate the d-miles from the map point to the camera's centre. If it is outside of the map point $d \notin [d_{min}, d_{max}]$ invariance zone, discard it.
4) Calculate the scale in the frame using the d/d_{min} ratio.
5) The map point is associated with the best match by comparing the representative descriptor **D** of the map point with the still unmatched ORB features in the frame, at the anticipated scale, and near **x**.

With all of the map locations in the frame, the camera posture has been completely optimized.

E. New Keyframe Decision

Choosing whether or not to generate the current frame as a new keyframe is a final step. We will strive to insert keyframes as quickly as feasible since it makes the tracking more resilient to demanding camera motions, often rotations, and because the local mapping has a way to cull unnecessary keyframes. The following prerequisites must all be satisfied in order to add a new keyframe:

1) The last global relocalization must have occurred more than 20 frames earlier.
2) Local mapping is inactive or the latest keyframe insertion was more than 20 frames ago.
3) At least 50 points are being tracked by the current frame.
4) The current frame trails K_{ref} by less than 90% of the points.

We enforce a minimal visual change rather than applying a distance requirement to other keyframes as PTAM (condition 4). Successful relocalization is guaranteed by condition 1, and decent tracking by condition 3. A signal is issued to suspend local bundle adjustment if a keyframe is entered while the local mapping is busy (second half of condition 2) so that it can process the new keyframe as quickly as feasible.

VII. Local Mapping

In this part, we go over the procedures the local mapping goes through with each new keyframe K_i.

A. KeyFrame Insertion

The covisibility graph is initially updated, with a new node added for K_i and the edges arising from the shared map points with other keyframes updated. The spanning tree connecting K_i to the keyframe with the most points in common is then updated. After that, we compute the keyframe's bag-of-words representation, which will aid in data association for triangulating new points.

B. Recent Map Points Culling

Map points must pass a stringent test within the first three keyframes following the creation in order to be kept in the map. This test verifies that the points are trackable and have not been incorrectly triangulated, that is, as a result of erroneous data association. A point must meet these two requirements:

1) More than 25% of the frames in which the point is expected to be visible must be found by the tracking.
2) Map points must be viewed from at least three keyframes if more than one keyframe has passed after they were created.

If a map point passes these criteria, it may only be deleted if it is seen from less than three keyframes at any one moment. Keyframe culling and local bundle modification that discards outlier observations can also result in this. Our map has extremely few outliers as a result of this strategy.

C. New Map Point Creation

By triangulating ORB from linked keyframes K_c in the covisibility graph, new map points are produced. We look for a match with another unmatched point in a different keyframe for every unmatched ORB in K_i. When this matching is finished, the matches that do not adhere to the epipolar restriction are discarded. After triangulating the ORB pairs, the scale consistency, reprojection error, parallax, and positive depth in both cameras are evaluated to accept the new points. A map point is initially seen from two keyframes, but as it could also match in other keyframes, it is projected into the other linked keyframes, where correspondences are then looked for.

D. Local Bundle Adjustment

The current processed keyframe K_i, every keyframe linked to it in the covisibility graph K_c, and every map point viewed by those keyframes are all optimized by the local BA. The optimization includes any other keyframes that view those points but are not related to the keyframe being processed at the time; yet, they all stay fixed. Both in the middle and at the conclusion of the optimization, observations that have been designated as outliers are eliminated.

E. Local Keyframe Culling

The local mapping seeks to find unnecessary keyframes and remove them in order to maintain a compact reconstruction. It also allows for lifetime operation in the same environment since the number of keyframes cannot increase indefinitely until the visual content of the scene changes. This is advantageous because bundle adjustment difficulty increases with the number of keyframes. Every keyframe in K_c that has 90% of the map points visible in at least another three keyframes of the same or finer size is discarded. The scale condition makes sure that map points have the keyframes that allow for the most accurate measurements.

VIII. Loop Closing

The loop closing thread uses K_i, the final keyframe that the local mapping processed, to search for and end loops. The process is then explained.

A. Loop Candidates Detection

The bag of words vector of K_i is compared to all of its neighbours in the covisibility graph first ($\theta_{min} = 30$), and the neighbour with the lowest score, s_{min}, is kept. Then we do a recognition database search and remove any keyframes with a score below s_{min}. While the normalizing score in DBoW2 is determined from the preceding picture, we leverage covisibility information in this process to increase robustness. Additionally, the results exclude any keyframes that are directly related to K_i. We need to find three loop candidates that are consistent in a row before we approve a loop candidate (keyframes connected in the covisibility graph). If there are numerous locations with K_i's appearance, then there may be several loop options.

B. Compute the Similarity Transformation

Three translations, three rotations, and a scaling factor are among the seven degrees of freedom in monocular SLAM that can cause the map to drift. Therefore, in order to terminate a loop, we must calculate a similarity transformation between the current keyframe K_i and the keyframe K_l of the loop, which tells us how much error has accumulated there. The loop's geometric validity will also be checked by the computation of this similarity.

We begin by calculating correspondences between the loop candidate keyframes and the ORB associated with the map points in the current keyframe. For each loop candidate at this time, we have 3D to 3D correspondences. With each contender, we alternately run RANSAC iterations in an effort to use Horn's approach to identify a similarity transformation. If a similarity S_{il} is discovered that has a sufficient number of inliers, we optimize it and conduct a guided search for more correspondences. Once further optimized, the loop containing K_l is approved if S_{il} is backed by sufficient inliers.

C. Loop Fusion

Fusing duplicate map points and adding new edges to the covisibility graph, which will attach the loop closure, is the first stage in the loop rectification process. In order to align both sides of the loop, the current keyframe posture T_{iw} is first adjusted using the similarity transformation S_{il}, and this adjustment is then propagated to all of K_i's neighbours. All map points viewed by the loop keyframe and its neighbours are projected onto K_i, and the area immediately around the projection is examined for matches. The matching map points and the inliers in S_{il}'s calculation are all combined. The edges of every keyframe involved in the fusion will be updated in the covisibility.

D. Essential Graph Optimization

We carry out a pose graph optimization over the essential graph that disperses the loop closure error over the graph in order to successfully close the loop. To rectify the scale drift, optimization is carried out through similarity transformations. Following optimization, each map point is modified in accordance with the adjustment made by a keyframe that is watching it.

IX. Experiments

We conducted a thorough experimental validation of our system using the large robot sequence from the TUM RGBD dataset to assess the system's overall performance. 16 hand-held indoor sequences from the TUM RGB-D benchmark were used to assess localization accuracy, relocalization, and lifelong capabilities. The input images of the dataset used are shown below:

Fig. 2: Input image for TUM rgbd dataset freiburg1 xyz dataset

A. Localization in the TUM RGB-D Benchmark

Our dataset includes a sequence of a camera moving across table, 20 frames per second and 640x480 resolution is used by a stereo camera to record the sequence. The sequence is particularly difficult for monocular vision since it has several loops and quick rotations. No other monocular system that we are aware of in the literature can handle the entire sequence. For instance, despite their ability to complete loops and work in large-scale settings. only demonstrated monocular findings for a short portion of this sequence. More than twice as many frames are correctly relocalized by ORB-SLAM as by PTAM. In the second experiment, we first create a map using the sequence $fr3_walking_xyz$ and then attempt to relocalize every frame from $fr3_walking_xyz$. The large occlusions caused by individuals moving around in the scene make this experiment

TABLE I: Computation Time on TUM RGB-D Dataset

Loop	KeyFrames	Essential Graph Edges	Total (s)
1	312	816	1.87
2	798	1574	4.23
3	967	2737	7.43
4	1243	4389	11.09
5	1467	6754	14.55

difficult. While our method relocalizes 73% of the frames in this instance, PTAM detects no relocalizations.

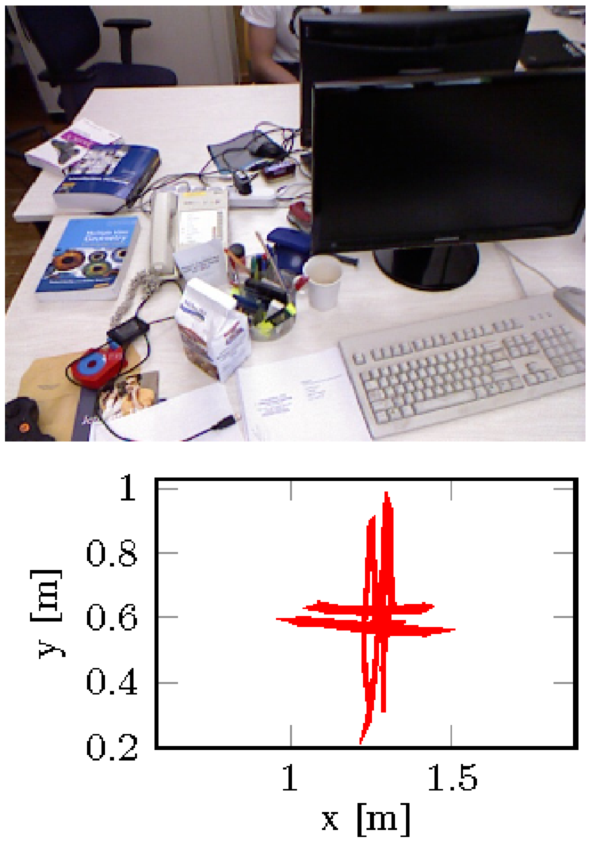

Fig. 3: Tracking output of TUM rgbd dataset xyz dataset

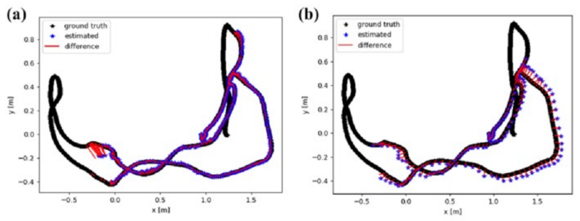

Fig. 4: Results on TUM dataset

B. Lifelong Experiment in the TUM RGB-D Benchmark

Our system can locate a map from a variety of angles and robustly under modest dynamic changes, according to previous relocalization studies. This trait, when combined with our keyframe culling technique, enables lifetime operation in the same environment while changing perspectives and experiencing certain dynamic changes.

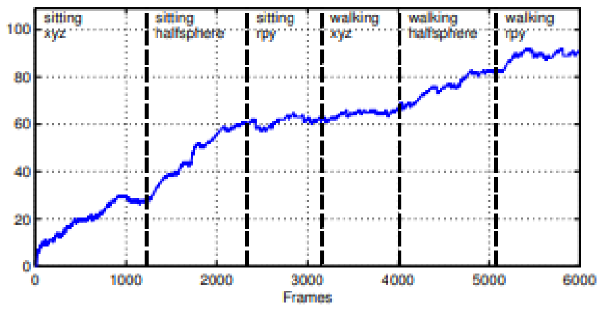

Fig. 5: Evolution of the number of keyframes in the map

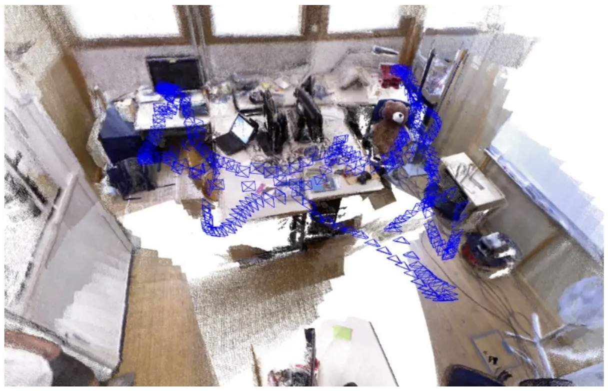

Fig. 6: Point Cloud of a room scene with 1 loop closure

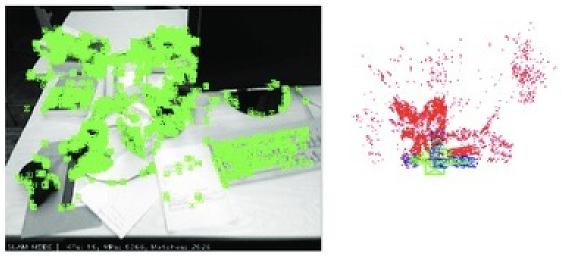

Fig. 7: The scenes of ORB-SLAM changed on TUM dataset $freiburg1\_xyz$

Even when the camera is observing the scene from various angles in a perfectly static scenario, our method is able to keep the number of keyframes constrained. We show it in a bespoke sequence where, for the entire 93 seconds, the camera is pointing at the same desk while moving along a trajectory to change angles. We contrast the changes in the number of keyframes produced by PTAM with our own map. It is clear that PTAM continuously adds keyframes, but our approach for removing unnecessary keyframes causes its number to saturate.

Any SLAM system should be able to operate continuously in a static setting, but the case where dynamic changes take place is more intriguing. By repeatedly performing the dynamic sequences from $fr3 \quad : \quad sitting\_xyz, sitting\_halfsphere, sitting\_rpy$, $walking\_xyz$, $walking\_halfsphere$, and $walking\_rpy$, we study the behaviour of our system in this circumstance. The same desk is the centre of every clip, but the camera moves in different directions as individuals move about and alter things like chairs. The map does not expand during the $fr3\_walking\_xyz$ and $fr3\_sitting\_xyz$ sequences since the current map adequately describes the situation.

In these long-term tests, we have demonstrated that our map increases with the scene's content but not with time. This ability to preserve the scene's dynamic changes may be valuable for performing some scene comprehension as a result of gaining exposure to an environment.

X. Conclusions

In this study, we introduced a novel monocular SLAM system, along with a thorough analysis of its constituent parts and public datasets. Our technology has proven that it is capable of processing sceneries from both indoor and outdoor settings as well as actions from cars, robots, and handheld devices. The system's precision generally ranges from a few meters in big outside settings to less than 1 centimetre in tiny inside scenarios (once we have aligned the scale with the ground truth).

Our work's key contribution is to increase PTAM's adaptability to conditions that are difficult for that system to operate in. The loop detection of Galvez-L'opez and Tardos [5], the loop closing procedure and covisibility graph of Strasdat et.al [6], [7], the optimization framework g2o by Kuemmerle et. al [37], and ORB features by Rubble et. al [9] are just a few of the excellent works developed in the last few years that we have incorporated into our new monocular SLAM. No other method has, to the best of our knowledge, proven to function as accurately and in as many distinct situations. As a result, our technology is now the most trustworthy and comprehensive monocular SLAM solution. Our unique keyframe spawning and culling procedure enable the creation of keyframes every few frames, which are then eliminated when they are deemed unnecessary. In weakly conditioned exploration trajectories, such as those that are nearly pure rotations or rapid movements, this flexible map expansion is quite helpful. The map only expands when a scene's visual content changes while running regularly in the same context, keeping a history of the scene's various visual incarnations. Analyzing this history may yield intriguing insights for long-term mapping.

Finally, we have shown that ORB characteristics have sufficient recognition strength to permit location recognition despite significant perspective changes. Additionally, they enable real-time precise tracking and mapping since they are so quick to extract and match (without the requirement for multi-threading or GPU acceleration).

XI. Future Work

Our system's accuracy can still be increased by including tracking sites at infinity. These spots, which do not have enough parallax and which our algorithm does not include in the map, provide a wealth of information about the camera rotation.

The upgrading of our system's sparse map to a denser and more practical reconstruction is another available option. Our keyframe selection has resulted in keyframes that provide a succinct overview of the environment with rich information about covisibility and extremely good pose accuracy. As a result, the ORB-SLAM sparse map can serve as an effective foundation upon which to build a dense and precise map of the scene.

Acknowledgment

We would like to thank Prof. Morato, our instructor for the RBE595: Motion Planning Sensor Fusion and Perception course at the Robotics Engineering Department of Worcester Polytechnic Institute for supporting our project during the Aug-Dec 2022 term. We owe our deep gratitude to him for taking a keen interest in our project and guiding us, by providing all the necessary information in and timely feedback during this project work.

References

[1] Masunga, Nsingi. "Mobile Robot Navigation in Indoor Environments by using the Odometer and Ultrasonic data." (1999).

[2] Gatesichapakorn, Sukkpranhachai et al. "ROS-based Autonomous Mobile Robot Navigation using 2D LiDAR and RGB-D Camera." 2019 First International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics (ICA-SYMP) (2019): 151-154.

[3] Biswas, Joydeep and Manuela M. Veloso. "Depth camera based indoor mobile robot localization and navigation." 2012 IEEE International Conference on Robotics and Automation (2012): 1697-1702.

[4] Weingarten, Jan W. and Roland Siegwart. "EKF-based 3D SLAM for structured environment reconstruction." 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (2005): 3834-3839.

[5] Cheng, Jiantong et al. "Compressed Unscented Kalman filter-based SLAM." 2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014) (2014): 1602-1607.

[6] Törnqvist, David et al. "Particle Filter SLAM with High Dimensional Vehicle Model." Journal of Intelligent and Robotic Systems 55 (2009): 249-266.

[7] Montemerlo, Michael et al. "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem." AAAI/IAAI (2002).

[8] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," IEEE Trans. Pattern Anal. Mach. Intell., vol. 29, no. 6, pp. 1052–1067, Jun. 2007.

[9] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in Proc. IEEE ACM Int. Symp. Mixed Augmented Reality, Nara, Japan, Nov. 2007, pp. 225–234.

[10] R. Mur-Artal, J. M. M. Montiel and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," in IEEE Transactions on Robotics, vol. 31, no. 5, pp. 1147-1163, Oct. 2015, doi: 10.1109/TRO.2015.2463671.

[11] Mur-Artal, Raul and Juan D. Tardós. "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras." IEEE Transactions on Robotics 33 (2017): 1255-1262.

[12] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. D. Tardos, ´ "A comparison of loop closing techniques in monocular SLAM," Robot. Auton. Syst., vol. 57, no. 12, pp. 1188–1197, 2009.

[13] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recog., New York, NY, USA, Jun. 2006, vol. 2, pp. 2161–2168.

[14] M. Cummins and P. Newman, “Appearance-only SLAM at large scale with FAB-MAP 2.0,” Int. J. Robot. Res., vol. 30, no. 9, pp. 1100–1123, 2011.

[15] D. Galvez-L ´ opez and J. D. Tard ´ os, “Bags of binary words for fast place ´ recognition in image sequences,” IEEE Trans. Robot., vol. 28, no. 5, pp. 1188–1197, Oct. 2012.

[16] Strasdat, Hauke et al. “Visual SLAM: Why filter?” Image Vision Comput. 30 (2012): 65-77.

[17] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige, “Double window optimisation for constant time visual SLAM,” in Proc. IEEE Int. Conf. Comput. Vision, Barcelona, Spain, Nov. 2011, pp. 2352–2359.

[18] H. Strasdat, J. M. M. Montiel, and A. J. Davison, “Scale drift-aware large scale monocular SLAM,” presented at the Proc. Robot.: Sci. Syst., Zaragoza, Spain, Jun. 2010.

[19] K. Pirker, M. Ruther, and H. Bischof, “CD SLAM-continuous localization and mapping in a dynamic world,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., San Francisco, CA, USA, Sep. 2011, pp. 3990–3997.

[20] S. Song, M. Chandraker, and C. C. Guest, “Parallel, real-time monocular visual odometry,” in Proc. IEEE Int. Conf. Robot. Autom., 2013, pp. 4698–4705.

[21] Myriam Servières, Valérie Renaudin, Alexis Dupuis, Nicolas Antigny, ”Visual and Visual-Inertial SLAM: State of the Art, Classification, and Experimental Benchmarking”, Journal of Sensors, vol. 2021, Article ID 2054828, 26 pages, 2021.

[22] M. Filipenko and I. Afanasyev, ”Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment,” 2018 International Conference on Intelligent Systems (IS), 2018, pp. 400-407, doi: 10.1109/IS.2018.8710464.

[23] Maity, Soumyadip, Arindam Saha, and Brojeshwar Bhowmick. ”Edge slam: Edge points based monocular visual slam.” Proceedings of the IEEE International Conference on Computer Vision Workshops. 2017.

[24] Newcombe, R.A.; Lovegrove, S.J.; Davison, A.J. DTAM: Dense tracking and mapping in real-time. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 2320–2327.

[25] Engel, Jakob, Thomas Schöps, and Daniel Cremers. ”LSD-SLAM: Large-scale direct monocular SLAM.” European conference on computer vision. Springer, Cham, 2014.

[26] Engel, Jakob, Vladlen Koltun, and Daniel Cremers. ”Direct sparse odometry.” IEEE transactions on pattern analysis and machine intelligence 40.3 (2017): 611-625.