An Investigation of Deletion Schemes for Dynamically Allocated Objects in ORBSLAM3

by

Nitin Vinod

9th January 2023

A thesis submitted to the
faculty of the Graduate School of
the University at Buffalo, The State University of New York
in partial fulfillment of the requirements for the
degree of

Master of Science
Department of Computer Science

To Maa, for the unconditional support.

Acknowledgements

My research on ORBSLAM3 would not have been possible without the steady help, guidance and patience from Dr Lukasz Ziarek and my colleague Shreyas. It was the long hours of discussion with both, and several experiments that helped me gain a good understanding of the aforementioned software system. I would also like to thank my parents for their unwavering support that enabled me to explore several areas of interest in the realm of computer science and keep moving forward.

Abstract

ORBSLAM3 is a simultaneous localization and mapping system that provides a high rate of accuracy compared to other SLAM systems available. It is significantly more robust in terms of image processing and has several new features ranging from multiple maps to various other optimizations. However, ORBSLAM3 does not have a scheme in place to free/reuse dynamically allocated memory. Dynamic memory allocation is used widely for storing image data/keyframes and marking 3D locations of features shared by several keyframes (also called map points).

The lack of such a scheme could potentially cause heap memory to be depleted provided the system runs for a long time, along with a huge number of memory leaks. The complexity in coming up with a viable scheme arises from the fact that most of the heap allocated memory within the system should be accessible by multiple threads at the same time.

As a result, predicting the lifetime of such objects is nontrivial. The deletion of such objects without any proper mechanism in place results in segmentation faults. This happens because the program encounters dangling pointers - references to previously freed heap allocated objects. This work outlines several experiments and methods that were used to come up with a proper deletion scheme that is memory safe. It also explores other possible solutions to the issue of reclaiming memory.

Table of Contents

List of Tables

List of Figures

Introduction

ORBSLAM3 is a simultaneous localization and mapping system that promises high accuracy compared to other SLAM systems. It is more resilient to a lack of visual information compared to its predecessors. The defining characteristics of ORBSLAM3 are namely - improved recall place recognition, an abstract camera representation, and a multimap atlas. The improvement in place recognition was achieved using a new algorithm that examined for geometrical consistency as well as local consistency.

ORBSLAM3 achieves state of the art performance in terms of SLAM but suffers from memory leaks due to its design. The multithreaded nature of the code as well as design decisions complicate the deletion of dynamically allocated objects. This work primarily focuses on the safe deletion of these objects allocated in the heap memory based on the design of the system.

Some of the other SLAM systems that preceded ORBSLAM3 include PTAM, LSD-SLAM, ORBSLAM, ORBSLAM VI and ORBSLAM2. A small overview of each of these SLAM systems is provided. The examination of different systemic components and how they interact with each other outlines the design and choices behind the construction of a SLAM system. Several of these systems have played an influential part in the design of ORBSLAM3, which explains the reasoning behind decisions related to the construction of the system.

The general outline of the document is as follows – a review of several SLAM systems based on their overall design, the insights based on the design of these systems as well as the issues associated with ORBSLAM3, and common methods used for deallocating memory. This is followed by a detailed account of several methods that were used to make an attempt to facilitate safe deletion for heap allocated objects in ORBSLAM3.

Related Work

PTAM

PTAM [1] stands for Parallel Tracking and Mapping. It is a visual SLAM system that completely disentangles the activities of the tracking process and the mapping process into different threads of execution. This meant that the majority of computations performed in each thread could be done without any synchronization with the other. There were other systems that incorporated both tracking and mapping, however both processes required a high degree of synchronization with each other.

The design of the PTAM system hypothesized that using two threads of execution for tracking and mapping could increase the performance of both the processes. More specifically, the tracking thread would be able to process the input from the camera sensor without any constraints pertaining to synchronization with the mapping thread.

In the case of mapping, not all frames created by tracking would require to be processed by the mapping thread. This would be beneficial as skipping frames that contain similar information would speed up the mapping process. Another design shift incorporated changing mapping to a batch method instead of explicitly mapping every single frame.

Tracking maintains track of the camera pose relative to the known environment and is roughly composed of capturing a new frame with the help of the camera, followed by figuring out and optimizing the pose of the camera.

The system evaluates the condition of the tracking process. If it is deemed weak, then the associated keyframes are not used for mapping procedure.

Mapping is the process by which a map is built from the features associated with the frames. Feature matching between various frames is used to create the map with the help of RANSAC [2] and FAST [3]. To insert new keyframes, the tracking process should be fairly successful with the input stream of images.

Bundle Adjustment is the batch method which is used in the mapping thread to optimize poses for keyframes. PTAM makes use of the Levenberg-Marquard [4] bundle adjustment implementation. Since tracking has stringent real time limitations, bundle adjustment oversees a small portion of feature detection.

The time taken to carry out the bundle adjustment procedure grows as the number of keyframes and map points within the map grows. There will be many new keyframes that have to be processed during the time bundle adjustment computations are done. As a result, the mapping thread would slow down. To prevent this issue, local bundle adjustment is carried out on a specific number of keyframes as opposed to performing computations on all of them.

Global bundle adjustment that is carried out for all the keyframes almost always takes up too much time beyond a certain number of keyframes. As a result, it often becomes impractical past a threshold and the computation is discontinued.

PTAM is compared to EKF-SLAM [5], a system in which the tracking and mapping processes are tightly coupled with each other. In a nutshell, PTAM is more resilient to quick movements of the camera sensor in terms of processing feature characteristics and provides a comparable quality of tracking and mapping.

DTAM

DTAM [6] is a SLAM system that uses pixel methods instead of feature extraction. Most existing SLAM systems use feature extraction for tracking, however dense methods may be used in real-time using consumer friendly GPUs. This is achievable as a large degree of computational operations may be run in parallel. The camera's motion is tracked in six degrees of freedom with the dense model created until that point. It can work in environments in which input stream of images incorporates rapid motion like PTAM.

The camera pose is optimized using Lucas-Kanade [7] nonlinear least squares in two steps. Even though the image processing pipeline strictly uses photometric methods, initialization is performed using a feature based stereo method. The tracking is modified to account for the pixels that do not have any tracking information associated with them.

DTAM performed well in the presence of unclear images as well as quick movements making it a promising approach for use in augmented reality and robotics. It is slightly more performant than PTAM in datasets with quick changes in velocity, leading to better tracking behavior. It also

exhibits a high degree of accuracy. However, DTAM is susceptible to changes in illumination, especially those changes that can occur in the real world.

LSD-SLAM

Large-Scale Direct SLAM [8] is a SLAM system that runs in real time using only CPU resources along with a monocular camera. It involves a direct tracking method and a probabilistic solution to account for the impact of depth values. It is aimed at use in augmented reality and robotics applications as it can run in real time.

Feature based methods are used to extract features from images and estimate camera pose. However, these methods are severely curtailed by a limitation of extractable information being highly correlated with the type of feature detection employed. More specifically, the usage of key points leads to some kinds of image data being unusable. Direct methods enable the usage of all the information that can be extracted from an image.

It also provides more information about characteristics such as the geometry of a particular scene with a high degree of accuracy. However, these methods are rather computationally expensive and require GPUs to be run in real time, but techniques used for semi dense visual odometry may be used at a lesser computational cost using only CPUs.

The rationale behind this approach is to combine the strengths of both feature based and direct methods, namely efficiency and robustness. Feature based monocular SLAM is limited to using specific feature types as well as feature matching while dense methods are held back by computational cost. A depth map is made for an image along with whole image alignment for

tracking and is used to figure out the camera pose for the next image. The system is initialized with the very first keyframe having an arbitrarily chosen depth map.

The SLAM algorithm has three salient elements – tracking, map optimization and depth map estimation. Tracking is responsible for capturing new images and figuring out the pose based on keyframes. The depth map estimation is used to improve the quality of processing the current keyframe and the map optimization element oversees the insertion of new keyframes into the preexisting map.

If a new keyframe is created, then the depth map associated with that particular keyframe is created by projection of points from the last keyframe. SLAM systems that make use of a single camera are susceptible to scale errors as opposed to stereo which leads to drift. LSD SLAM attempts to alleviate this issue by using the relationship between tracking accuracy and scene depth. Images are aligned with the help of the weighted Gauss-Newton optimization method.

Map optimization is performed using optimization of pose graphs. This is facilitated with the help of the g2o [9] framework that is used in many SLAM systems. Upon evaluation, the system was found to be capable of robustly performing tracking in difficult datasets belonging RGB-D [10] dataset suite.

ORBSLAM

ORB [11] is used for feature detection in ORBSLAM [12], it is computationally more efficient compared to algorithms used for feature detections such as SIFT [13]. However, it also has drawbacks, it is not as performant as SIFT as well as SURF [14] in certain scenarios.

The rationale behind creating the ORB key point detector and descriptor was to have devices with lower computational power be capable of various image processing techniques as well as decrease the time for feature detection. The ORB key point detector and descriptor makes use of the FAST key point detector as well as BRIEF [15] descriptor.

The detector determines the major features of an image, while the descriptor deals with properties of the image such as scale as well as orientation. FAST stands for Features from Accelerated Segment, the main takeaway being that it is computationally more efficient compared to other descriptors.

BRIEF contributes heavily to ORB by having extremely low rotational invariance. This makes ORB very suitable for real time operation within embedded devices. ORB is faster than both SIFT as well as SURF by at least one order of magnitude.

ORBSLAM [12] makes use of a covisibility graph which enables the system to work in real time. The system is capable of real time loop closing and camera relocalization as well. It also incorporates a strict policy for the creation and culling of map points as well as keyframes. Compared to PTAM, ORBSLAM allows the creation of many keyframes so that

tracking is possible in tough situations. PTAM was designed to be conservative with the creation of keyframes due to issues rising from the complexity of processing many keyframes.

In ORBSLAM, the three main threads are to facilitate the tracking process, loop closing process and the mapping process. Tracking is responsible for estimating the camera pose and figuring out when to create a new keyframe. Feature matching is carried out along with motion only bundle adjustment. Camera poses are calculated again by using reprojection on every map point.

The mapping thread performs local bundle adjustment, it also makes sure to mark keyframes and map points that need not be used anymore. Even though a scheme exists to mark keyframes and map points, a mechanism to facilitate their safe deletion does not exist. Both keyframes as well as map points are heap allocated objects and consume quite a bit of heap memory.

This increases with running time due to ORBSLAM's policy of creating many keyframes in the beginning while tracking, followed by removing the ones not required. Lastly, the loop closing thread attempts to figure out whether the past keyframes and current keyframe are related via means of a loop. The covisibility graph is undirected and conveys the relationship between two keyframes. A node represents a keyframe and the edge of the graph between two nodes will be present if they have common features from the same map points. To facilitate loop correction, pose graph optimization is accomplished using an essential graph with a smaller number of edges.

The essential graph contains a spanning tree which is updated with when keyframes are taken out and added. The optimization related to the pose graphs is performant to the degree that carrying

out bundle adjustment again is hardly necessary. Loop detection and relocalization for the ORBSLAM system is carried out using DBoW2 [16].

The camera pose is calculated, however if estimation fails, a larger number of map points are searched. If the system is unable to track, RANSAC [2] along with the PnP algorithm [17] is used to come up with a pose. Once the system arrives at a proper estimate of pose, map points present in the local map are projected on the keyframe to optimize pose.

This is to make sure that the pose is calculated with respect to each and every map point within the frame. Finally, the decision to insert a keyframe must be taken, this requires that a certain number of frames should have passed the last keyframe insertion, and that current frame tracks successfully to a certain extent.

Within the mapping thread, the addition of a new keyframe causes the covisibility graph to be adjusted. This causes the spanning tree to be updated as well. The culling policy for keyframes as well as map points is structured around the mapping thread. To cull a map point, it must be present in more than 25% of the keyframes, also it must be observable from at least three keyframes if the system processes more than one keyframe since the map point was created. New map points are created based on finding new features within keyframes.

The keyframe culling policy mainly figures out the extraneous keyframes and flags them appropriately. This is mainly carried out to reduce computational costs associated with bundle adjustment as the number of map points and keyframes within the map keeps growing. If a lot of

the map points associated with a keyframe are seen in a couple of other keyframes, then the keyframe is marked bad. The tests conducted with ORBSLAM indicate that tracking is the most significant task in terms of time taken to process map related information.

Similarly, for local mapping, the bundle adjustment takes quite a bit of time. The main advantage of ORBSLAM compared to other systems like PTAM is the performance and accuracy exhibited by the system across different types of environments and settings involving quick motion from one spot to another as well as motion with a rotational component. There are other SLAM systems that incorporate dense methods to create the environment. However, they are limited by a wide range of problems such as artifacts related to rolling shutter, auto gain, and auto exposure.

BASALT

BASALT [18] is a SLAM system that aims to use data from visual inertial odometry to aid in visual inertial mapping; it uses input data from inertial measurement units as well as an image sensor. Bundle adjustment, if performed with data from the inertial measurement unit would yield accurate and useful results but at a high computational cost.

Instead of doing so, BASALT aims to figure out nonlinear factors [19] which may be used to estimate camera motion and use these for global bundle adjustment. Simultaneous localization and mapping based methods use graph-based optimization which may be prone to having error prone results due to operating on only a subset of the variables involved.

BASALT estimates nonlinear factors in terms of the Kullback-Leibler divergence [20], which aids in the creation of higher quality pose estimates. Key points for the image are detected with the FAST corner detector. For global map optimization, ORB features are detected amongst keyframes. This is used in tandem with the nonlinear factors for optimization. The system is evaluated using the EuRoC [21] dataset. In terms of mapping, the system is compared to ORBSLAM.

Both the systems make use ORB features, with the distinguishing difference being the use of nonlinear factors used by the former and preintegrated measurements amongst for keyframes by the former. In terms of performance, BASALT does considerably better than ORBSLAM for a couple of datasets that involve a large environment.

The difference in performance reduces when keyframes are created within a shorter period of time as opposed to longer durations of time in larger environments. BASALT's defining feature is the minimized computational cost for optimization using nonlinear factors.

ORBSLAM2

ORBSLAM2 [22] makes use of RGB-D cameras as well as stereo cameras. This enables the system to gauge depth from the stereo key points to start the creation of the map. Motion only bundle adjustment is carried out in the tracking thread. During the loop closing operation in ORBSLAM, there is a good chance of scale drift, which is highly unlikely for ORBSLAM2, given the presence of stereo image input. The system has an added policy for deciding when to create a new keyframe based on the proximity of the stereo camera sensor to the surroundings.

ORBSLAM2 was tested primarily using the KITTI [23], EuRoC [21] and the TUM RGB-D [10] datasets. In the KITTI dataset, stereo enabled ORBSLAM2 can function properly where monocular ORBSLAM failed. This boils down to having more information, given the presence of the stereo sensor. ORBSLAM2 was more performant than LSD SLAM and had a lower error rate.

It was evaluated on the EuRoC dataset which consisted of images from a micro aerial vehicle moving in several rooms. This was carried out in different amounts of lighting and velocities. The system is unable to track in some cases but obtains a higher accuracy than LSD-SLAM. There is a policy to mark keyframes and map points bad, however there is no safe deletion scheme to free the keyframes and map points that are no longer required.

ORBSLAM VI

ORBSLAM VI [24] is a visual inertial SLAM system which is a successor to ORBSLAM. It is capable of closing loops and able to make use of premapped areas to improve drift. It makes use of a covisibility graph for keyframes and map points along with pose graph optimization for loop closure. The system tracks the current keyframe and performs bundle adjustment in a different thread of execution. An initial bundle adjustment is required to optimize the local BA carried out by the mapping thread. This requires a high-quality initial solution.

This solution is computed using a divide and conquer approach, which involves processing the first couple of seconds of the input video, estimating the bias of the gyroscope using keyframe

orientation as well as figuring out the bias associated with the accelerometer. The system may also use localization only mode in case of a constraint relating computing resources.

The initial solution for the bundle adjustment only requires that two successive keyframes are not temporally far apart. More specifically, the estimation of the solution may be split into gyroscope estimation bias, gravity approximation without accounting for accelerometer bias, accelerometer bias estimation, scale direction refinement as well as velocity estimation.

This system is also composed of three unique threads namely tracking, loop closing as well as local mapping. The pose of the camera sensor is estimated in tracking along with IMU biases, and the key points in the frame are matched with map points that are obtained from the local map via means of projection. The Gauss Newton algorithm from the g2o [9] library is used to solve optimization problems related to tracking a keyframe immediately after an element is added to the map.

The local mapping thread oversees full as well as local bundle adjustment, it makes use of a specific quantity of keyframes before the current keyframe. This approach makes sure that it also marks the redundant keyframes which are no longer useful.

The loop closing error reduces the drift with the help of pose graph optimization. The pose graph optimization is carried out on 6 degrees of freedom as opposed to 7 in ORBSLAM. It does not use any information from the inertial measurement unit.

The system is evaluated using the EuRoC [21] dataset, with local bundle adjustment set to process 10 keyframes. The dataset consists of 11 sequences captured by a micro aerial vehicle moving in a couple of environments. The data is demarcated into three divisions based on inherent characteristics such as motion blur, texture and illumination.

The evaluation metrics indicate that the system is robust enough to process most sequences in real time but fails while processing certain datasets where the degree of motion is too much for the system to work properly. However, the system is more robust when used in its visual inertial setting but with a higher computational cost for bundle adjustment.

The main improvement that this system exhibits is zero drift accumulation, which is a pervasive issue in systems that use only visual odometry. This system may be tailored towards use in applications that use augmented reality. The system may have better performance if used with stereo cameras, this will also make the IMU initialization a far easier task.

OKVIS and ROVIO

OKVIS [25] is a SLAM system that makes use of both inertial information along with visual information. The system is tightly coupled with respect to its input systems; it uses both inertial and visual information jointly. It uses nonlinear estimation along with other methods to reduce computational complexity. The image processing is carried out with the help of Harris corner detectors [26] combined with BRISK descriptors.

ROVIO [27] stands for Robust Visual Inertial Odometry which is a SLAM system written in C++, which makes use of an extended Kalman filter, and tight coupling between visual inertial odometry and a photometric error model. It uses direct approaches like DTAM mentioned above.

It was also tested with the EuRoC [21] datasets, with better performance in the second half of datasets. It exhibits a marked difference in performance for the stereo and monocular configurations. The performance for this system was on par with OKVIS, with slightly better execution times for smaller distances.

VINS-Mono

VINS-Mono[18] is a SLAM system that makes use of visual as well as inertial data in order to estimate state. The salient features are as follows: an initialization mechanism to start the system, a tightly coupled visual odometry system with IMU bias correction, four degrees of freedom pose graph optimization and reuse of the pose graph. The most common way to couple together visual and inertial data is to make use of sensor fusion using the extended Kalman filter. Tightly coupled visual inertial systems use either the extended Kalman filter [28] or optimization of the pose graph.

Visual and inertial measurements are carried out between the present frame and its successor, at the same time, inertial measurements unit readings are preintegrated. For tracking, new corners are determined using the Shi Tomasi [29] corner detector while features detected earlier in other keyframes are estimated/processed using the KLT sparse optical flow algorithm [30]. RANSAC [2] is used to get rid of redundant data.

There are two constraints used for the selection of new keyframes, these are on the basis of the parallax of tracked features and the quality of tracking. A new frame will only be deemed a keyframe if a certain count of features corresponds with the past keyframe. Gyroscope measurements are used for parallax calculations to accommodate the error caused due to rotation. The nature of the tightly coupled inertial visual system is quite nonlinear, thereby requiring an initial seed.

Visual inertial alignment is responsible for the synchronization of the visual data and the readouts given by the inertial measurement unit. DBow2 [16] is used for loop detection, like the ORBSLAM systems. VINS was tested and evaluated on the EuRoC dataset.

VINS performs with low error rate for loop closure in most of the datasets compared to OKVIS and was also quite performant for the map merging operation. The system was quite performant with very less drift for a long period of time. VINS exhibits complete removal of drift due to the 4 DOF optimization of the pose graph.

ORBSLAM3

ORBSLAM3 [31] is the successor to ORBSLAM2 and ORBSLAM-VI with a better recall place recognition, monocular as well as stereo visual inertial SLAM, an atlas – to support multiple maps that are not directly connected. This SLAM system makes use of techniques that can take full advantage of short term, midterm as well as long term data association. Short term data association makes use of only the map components obtained in the last couple of seconds.

Midterm data association matches components that have a small drift and long-term data association allows to match components in areas that were previously visited. Map optimizations could be carried out with the help of bundle adjustment. This system provides multimap data association and has quite a bit of changes compared to ORBSLAM2 pertaining to the atlas – a combination of disconnected maps.

The tracking thread carries out the same function as in ORBSLAM2 but if lost tries to relocalize with respect to all the maps in the atlas as opposed to just one map in ORBSLAM2. The mapping thread performs the same function of adding keyframes and marking them bad.

The loop closing along with map merging thread attempts to figure out the common areas between the map at hand and the set of map present within the atlas. Bundle Adjustment may also be performed if required. The initialization performed for the IMU sensor is quicker than that of ORBSLAM-VI.

The experimental results indicate that ORBSLAM3 outperforms most of its competitors such as ROVIO, VINS-Mono and KIMERA. It does better than ORBSLAM-VI in terms of accuracy due to having a better initialization mechanism for IMUs. The multimap system would contribute to increasing the robustness of the system if only the visual mode were to be used. This may be used to navigate through difficult datasets that posed a challenge to ORBSLAM2. Visual inertial systems perform a lower number of loop closures compared to purely visual systems. This behavior is because the latter will have more drift compared to the former.

The system is highly performant in the TUM-VI [32] dataset, two other systems that are comparable are VINS-MONO and BASALT. ORBSLAM3 has the least error rate when used in indoor scenes in which previous scenes are almost always visited. However, VINS-Mono and BASALT can track better in some environments with very poor visual information. It has almost double the accuracy compared to its competitors in single session SLAM owing to map merging.

ORBSLAM3 exhibits poor performance when it comes to scenes and environments with low texture, which is better handled by the methods like Lucas-Kanade [30] used by BASALT. Amongst the main sensor configurations namely - monocular, stereo, monocular-inertial and stereo inertial, the latter has the best performance in terms of accuracy.

KIMERA

Kimera [33] is a SLAM system written in C++ which makes use of metric semantic understanding. Metric semantic understanding involves the mechanisms to associate markers for objects present in a scene as well as provide visual context.

Kimera mainly makes use of 4 modules namely KIMERA-VIO, KIMERA-RGPO, KIMERA-Mesher as well as KIMERA semantics. The VIO module is used for visual inertial odometry, RPGO module is used for pose graph optimization, the Mesher module helps with obstacle avoidance by means of using a 3D mesh. The semantics module optimizes the 3D mesh created by the Mesher.

The novelty behind Kimera comes from the use of semantic labels as well as the use of modules for different functions. Kimera's processing can be mapped to the execution of 4 threads in correspondence to the modules stated above. The visual inertial odometry module along with its associated thread is responsible for the capture of stereo images, processing the input of the inertial measurement unit and providing the estimates for the unit.

The second thread is also associated with the VIO module, as well as the Mesher module. It improves on the estimates provided by the first thread and creates meshes. The threads mentioned above mainly deal with the initial processing of input data as well as mesh creation. The third thread deals with graph optimization and loop closure. The fourth thread is associated with the semantics module, which oversees semantic segmentation.

The VIO module makes use of the a posteriori estimator that is also used by some of the SLAM systems mentioned earlier. RANSAC [2] is used for feature detection in both stereo as well as monocular modes. Tracking is not carried out across all the keyframes but only across some of them.

The RPGO module is primarily used for the detection of loops amongst keyframes, it makes use of DBoW2 [16]. The mesher creates two types of meshes namely a per frame mesh as well as a multi frame mesh. Delaunay triangulation [34] is extensively used for the creation of the mesh. The multi mesh is created by merging many singular frames.

The semantics module makes use of bundled raycasting to create a mesh that maps the entire path as well as provide semantic segmentation. The module makes use of Voxblox [35], a volumetric mapping library that makes use of truncated signed distance fields and adapts it to create the mesh.

The mesh is semantically marked using 2D images that are produced from keyframes. Kimera provides tools to evaluate the performance of the VIO as well as the RPGO module. It benchmarks the module on EuRoC [21] datasets. Kimera exhibits very good performance in terms of pose estimation and geometric reconstruction.

The semantic reconstruction was tested using a Unity based simulator which provided ground truth for a dataset. The performance is most impacted by dense stereo which is used to compute a 3D point cloud from a pair of stereo images. This matters most when it is difficult to estimate the depth of scenes including walls and similar artifacts.

OKVIS2

OKVIS2 [36] is a SLAM system that can deal with long and repeated loop closures, it makes use of a convolutional neural network to process keyframes. The convolutional neural network is run on a CPU to facilitate portability. The system is composed of a front end and a real time estimator to deal with processing input data.

More specifically, the front end is responsible for handling key point matching, place recognition and loop closing. It is also responsible for the convolutional neural network. The real time estimator oversees pose graph construction as well as factor graph optimization.

BRISK [37] key points and descriptors are captured from the input images, which are later matched. BRISK is an alternative to both SIFT [13] and SURF [14]. SIFT has quality results but comes at a high computational cost. The FAST [3] as well as the BRIEF [15] key point descriptors are suitable for real time applications but are susceptible to image distortions.

BRISK descriptors are matched using Hamming distance [38] and make use of a circular sampling pattern to estimate brightness. It offers much more resilience to rotations and is scale invariant. BRISK has comparable performance to SIFT and SURF. It is best suited for real time tasks with a hard constraint for computational power.

The decision to incorporate a new keyframe is based on whether there are matches in the current frame with all the previous keyframes. The convolutional neural network was previously trained on datasets before running the SLAM system. In terms of performance, OKVIS2 has comparable performance to ORBSLAM3 [31] in some of the datasets. The salient features of OKVIS2 include a factor graph of errors as well as loop optimization.

Design Insights

A couple of generalized insights may be gleaned from the overview of the SLAM systems above. Firstly, the computational load is mostly split into different threads of execution to speed up computation. Each thread would oversee tracking, local mapping, loop closing or another required task. Secondly, most systems involve sharing resources among threads to facilitate faster computation.

Thirdly, low resource usage would aid a SLAM system. Most of the SLAM systems use similar constraints to create a new keyframe and involve some form of bundle adjustment and pose graph optimization. Most of them use feature extraction while only some use direct methods for tracking.

All the observations stated above are in line with the design of ORBSLAM3. The ORBSLAM family of SLAM systems performs SLAM operations very well but suffers from a variety of memory leaks caused by not freeing dynamically allocated memory after usage. This is mostly in the form of keyframes and map points, as well as objects used for optimization computations.

The structure of keyframes and map points are such that each keyframe can hold references to many map points. Similarly, a map point can hold many references to many keyframes. Map points are not unique to keyframes and vice versa. Along with this cyclic property, the pointers to these keyframes and map points are passed to 4 threads of execution making it difficult to determine when it would be safe to delete an object. Since ORBSLAM is written in C++, premature deletion would cause a segmentation fault bringing the entire system to a standstill.

ORBSLAM, ORBSLAM2 and ORBSLAM3 all have mechanisms to mark keyframes and map points bad, however these mechanisms merely mean that the SLAM algorithm does not have any use for the flagged objects. None of these systems have a mechanism where the flagged objects are deleted.

A two-minute run through an environment roughly generates about 250-300 keyframes and about 10,000 map points. If the program were to run for a longer time, the lack of memory reclamation would be inefficient. The keyframes and map points are shared amongst threads by using raw pointers, the interactions between threads make it quite difficult to safely delete objects. In short, the scale at which objects are allocated and complexity of the system makes it difficult to predict lifetimes for these objects.

Memory Deallocation

There are two well-known methods to reclaim heap memory: reference counting and garbage collection using a collector. Reference counting [39] is a classical garbage collection algorithm that associates a count with each object that is created in the heap memory. The count for each object is the total number of references that other objects hold for this particular object at a given point in time. A new reference to the object would increment the count by one while the removal of a reference would cause a decrement in the count by one.

A count of zero would mean that the object is ready to be freed; for a language like C++ this would entail calling the destructor of the associated heap allocated object which would use the delete

keyword. Reference counting enables memory to be freed right as the count drops to zero, this behavior makes it well suited for memory critical applications [39].

However, this method of memory reclamation if used naively is susceptible to leaking memory if used with cyclic structures. Reference counting does require a space overhead for storing a count for each object. It also facilitates the need for constantly updating the reference counts associated with each object [39]s.

Garbage collectors work by traversing the heap memory and marking all the live objects. It then proceeds to collect all the objects that are not alive. However, garbage collectors require good support from the runtime to keep track of the roots. Cyclicity would not be an issue for a tracing garbage collector [39].

C++ attempts to provide memory management using reference counting in the form of shared pointers. In case of cyclicity, weak pointers may be used be used to break the cycle. However, finding the correct positions to break the cycles using weak pointers is not trivial.

Languages like PHP, Objective-C, and Swift also use reference counting, support for atomic reference counting is also provided. There are also conservative garbage collectors available for C++ such as the Boehm Weiser Demers garbage collector [40]. However, this would require some changes to the infrastructure used by the codebase and wouldn’t be as simple as using a garbage collected language.

Memory Deallocation Schemes

Several methods were employed to figure out when objects could be safely deleted. Primary methods were based on the state of the mbBad flag for keyframes and map points. This was set to true by the system after it deemed that those keyframes and map points had no intrinsic value for SLAM operations.

Collecting the objects and deleting them after the system deemed them bad resulted in segmentation faults. Adding delays of varying time periods between the time points of marking the object bad and deleting them also resulted in segmentation faults.

The multithreaded nature of the system made it very difficult to avoid segmentation faults. The act of marking a keyframe or map point bad is carried out by the local mapping thread while the loop closing or tracking thread would be performing a computation using the pointer to that keyframe or map point. In some cases, several functions used for optimizations would hold on to the pointers of objects marked bad and use them with checks to avoid using redundant data.

The retention of stale keyframes and map points within the system even after the objects had been marked bad, meant that deletion only be safely carried out after the stale data had been flushed out of the system. A detailed explanation of the methods used to attempt memory reclamation in ORBSLAM3 is provided below.

Shared Pointers

The next step taken involved the use of smart pointers offered by C++ under the Resource Acquisition is Initialization (RAII) idiom. The shared pointer class [41], a type of smart pointer was used to provide handle all the raw pointers associated with keyframes and map points. The shared pointer keeps a count of its associated resource in the heap memory with the help of the copy constructor and invokes the destructor for deletion when the count for the object reaches zero.

More specifically, all the pointers pertaining to keyframes, and map points were replaced by their shared pointer implementation. In ORBSLAM3, the SetBadFlag is the function that marks a keyframe or map point bad based on its redundancy.

It also attempts to clean up all the references to the object in question, from other keyframes and map points. Ideally all the references should have been cleaned up and deletion should have occurred, but this scheme did not work as the destructor for neither the keyframe nor the map point was called.

The likely culprit for this behavior might be the cyclic nature of both keyframes and map points which held multiple references to each other. A plausible explanation is that references to the object marked bad were added to other keyframes and map points even after the removal operation.

This behavior is highly likely as references to objects marked bad still circulate the system until they are flushed by the clearing and rewriting of references from several containers. It also may be that pointers are stored in some containers that are hardly used or cleared.

In this case, the count for the marked object would never reach zero because some other object possessed a reference to it. The reads and writes of references to containers present in other keyframes and map points are quite difficult to track due to concurrency and the number of such objects used in the system. The large number of objects also makes it quite difficult to resolve the issue with the help of weak pointers/break the cycle.

Garbage Collection with the Boehm Demers Weiser Collector

The Boehm Demers Weiser Collector [40] is a garbage collector originally created for C and C++. It is capable of functioning incrementally and supports multiple threads. However, it is a conservative garbage collector. This means that there could be pointer misidentification leading to memory leaks.

The garbage collector was compiled and exposed to ORBSLAM3 with the help of a shared library. This involved changing all the heap allocations in ORBSLAM3 to stop using the keyword new to instantiate new classes in heap memory. The new keyword was replaced by its GC supported counterpart. This was followed by the removal of the delete keyword that freed memory for objects.

The use of the garbage collector led to segmentation faults quite frequently. Debugging sessions revealed that this was due to several manual memory allocations and deallocations happening within OpenCV, the library used for image manipulation within ORBSLAM3.

Using the garbage collector successfully might possibly entail modifying several dependencies of the system. Apart from OpenCV [42], there are also several other dependencies such as DBoW2 [16] and Eigen [43] used for mathematical operations and optimizations which might have to be modified for the GC to function properly.

Thread Safe Reference Counting with Mutexes

Reference counting with thread safety was implemented into certain sections of the ORBSLAM3 codebase to investigate whether deletion was possible. Due to the expansive nature of the codebase, reference counts were implemented on containers present in keyframes, map points and other locations based on requirement. The containers used would normally be vectors, sets, hash maps or linked lists. These containers normally held other references to other keyframes and map points in the form of raw pointers.

This approach enabled proper testing of the instrumented functions. This was a necessity due to the scale of the codebase. Locally tested functions with verified counts could be used to validate the count rather than doing the same for the entire codebase. While debugging, this form of counting helped to gather information about missing and extra counts on a function-to-function basis.

Verified counts for a function could be checked by tracking the state of the count for all the necessary objects at the beginning and end of the function call. Once a group of these functions were validated, the reference counts were guaranteed to be correct.

In terms of keyframes and map points, these functions mostly took the form of a get call which would pass all the keyframes and map points pointers associated with an object to the caller. As per the scheme mentioned above, each keyframe and map point was modified to have a reference count associated with its major containers.

The associated count would be incremented during a get call and decremented after the caller function finished its operations. The count for each of the containers had to be protected by a mutex owing to the multithreaded nature of the program. Each of these containers could be accessed by either the tracking, local mapping, loop closing thread or viewing thread. Hence, the count had to be protected from data races.

The following containers belonging to the keyframe object were instrumented – mConnectedKeyFrameWeights, mvpMapPoints, mConnectedKeyFrameWeights, and mvpOrderedConnectedKeyFrames. The functions covered were AddConnection, UpdateBestCovisibles, GetVectorCovisibleKeyFrames, GetBestCovisibilityKeyFrames, GetCovisiblesByWeight, AddMapPoint, EraseMapPointMatch, ReplaceMapPointMatch, UpdateConnections, SetBadFlag and EraseConnection.

Only the mObservations container belonging to the mappoint object was instrumented. The functions that were instrumented for the mappoint object are AddObservation, EraseObservation, GetObservation, SetBadFlag, Replace and isBad.

Apart from the reference counting mentioned above, the loop closing, local mapping and tracking objects used by their respective threads were also instrumented with reference counts. Reference counts were also placed inside functions for the following objects – Map, MapDrawer, and KeyFrameDatabase.

A tabular representation of the containers and functions mentioned above along with their instrumented containers is depicted below.

Table 1 KeyFrame Instrumentation

Containers in the KeyFrame object	Functions in the KeyFrame object
mConnectedKeyFrameWeights	AddConnection
mvpMapPoints	UpdateBestCovisibles
mConnectedKeyFrameWeights	GetVectorCovisibleKeyFrames
mvpOrderedConnectedKeyFrames	GetBestCovisibilityKeyFrames
	GetCovisiblesByWeight
	AddMapPoint
	EraseMapPointMatch
	ReplaceMapPointMatch
	UpdateConnections
	SetBadFlag
	isBad

Table 2 MapDrawer Instrumentation

Containers in the MapDrawer object	Functions in the MapDrawer object
	DrawKeyFrames
	DrawMapPoints

Table 3 MapPoint Instrumentation

Containers in the MapPoint object	Functions in the MapPoint object
mObservations	AddObservation
	EraseObservation
	GetObservation
	SetBadFlag
	isBad

Table 4 Map Instrumentation

Containers in the Map object	Functions in the Map object
mspKeyFrames	AddMapPoint
mspMapPoints	EraseMapPoint
mvpReferenceMapPoints	SetReferenceMapPoints

	GetAllMapPoints
	GetReferenceMapPoints

Table 5 KeyFrameDatabase Instrumentation

Containers in the KeyFrameDatabase object	Functions in the the KeyFrameDatabase object
	DetectNBestCandidates

Table 6 Optimizer Instrumentation

Containers in the Optimizer object	Functions in the Optimizer object
	GlobalBundleAdjustemnt
	LocalBundleAdjustment

Table 7 Loop Closing Instrumentation

Containers in the LoopClosing object	Functions in the LoopClosing object
	NewDetectCommonRegions
	DetectCommonRegionsFromBoW

Table 8 Local Mapping Instrumentation

Containers in the LocalMapping object	Functions in the LocalMapping object
	MapPointCulling
	CreateNewMapPoints
	SearchInNeighbors
	KeyFrameCulling

Table 9 Tracking Instrumentation

Containers in the Tracking object	Functions in the Tracking object
	Track
	CreateReplacedInLastFrame
	UpdateLastFrame
	UpdateLocalPoints
	UpdateLocalKeyFrames

The reference counting was done incrementally at different stages for all these objects. If a segmentation fault during a test run with a dataset, the point of failure would be found and the function in question would have the necessary keyframes/map points reference counted.

Thread Safe Reference Counting with Compare and Swap

Compare and Swap is an atomic instruction that carries out the comparison of an integer with a specified integer followed by the modification of the specified integer within memory. If the comparison is evaluated to be true, the swap is carried out. If false, no swap is carried out. Once the operation starts, the entire sequence of steps mentioned above is guaranteed to happen without any interruption. This should make reference counting faster.

Atomic operations are used to build lock free algorithms where synchronization is carried out with instructions like CAS as opposed to using a mutex. In terms of reference counting, the increment and decrement operations are implemented using compare and swap. This is facilitated with the help of a while loop.

Once a thread enters the critical section after the compare operation is successful, no other thread will enter the critical section before the count is incremented or decremented. Instead, the other thread will have to retry until it gets the current value of the count and perform a compare successfully to enter the critical section. This construct ensures the proper functioning of reference counting.

C++ provides atomic_compare_exchange_strong [44] from its standard library which enables the reference count to be implemented with CAS. The codebase was modified to have atomic integers used for reference counting instead of normal integers. The mutex locked counts within the functions were swapped out to incorporate while based CAS loops. Ifdefs were used to facilitate using reference counts with mutexes and reference counts with CAS loops.

The rationale for using compare and swap was to increase the speed of reference counting by avoiding putting threads to sleep and blocking them for long periods of time.

Thread Safe Reference Counting with Thread Based Counts

The main idea behind this implementation was to get rid of the contention for the variable holding the reference count. Instead of having a singular reference count for a keyframe or a map point, each object would have a count associated with a particular thread. For the actual reference count, the values of all the thread-based counts would have to be added up and checked. This should enable reference counting to work faster.

During the execution of any get calls from separate threads for the containers mentioned above, the system would not have to wait for locks to release to increment the reference count. The same would apply for decrements. In theory, this could be applied to any of the reference counting operations throughout the codebase.

The number of threads used by ORBSLAM3 is not dynamic, most of them are initialized during the system startup. The thread identification class objects for each thread were collected and stored

into each keyframe and map point right after initialization. For this, the keyframe and map point objects had to be modified to accommodate the thread identification objects.

Since each thread had to be associated with a count, a mapping between thread identification objects and counts was created with the help of a hash map within keyframes as well as map points. The hash map is composed of key value pairs where the key is the thread id object, and the value is the reference count associated with a particular thread id object.

Each thread could access the hash map in the keyframe or map point and modify the right count using this_thread::get_id() as the key. The identification objects for the tracking, local mapping, loop closing and viewing threads were inserted into the above specified hash map. All the mutex based counts within the system were replaced by the scheme mentioned above.

Experimental Results

ORBSLAM3 Statistics Without Memory Deallocation

ORBSLAM3 was profiled on the EuRoC dataset with an Intel i7-11857G with 32 gigabytes of memory. The EuRoC dataset consists of 11 sequences namely – Machine Hall 01, Machine Hall 02, Machine Hall 03, Machine Hall 04, Machine Hall 05, Vicon Room 1 01, Vicon Room 1 02, Vicon Room 1 03, Vicon Room 2 01, Vicon Room 2 02, Vicon Room 2 03. The times for measurements pertaining to function execution times are in milliseconds.

The calculations for all quantities are averaged across 10 iterations of each sequence. The following tables represent the number of keyframes, and map points processed and marked for deletion. Execution times of the local mapping, loop closing, and the tracking thread are also tabulated below. These statistics were captured to evaluate both the reference counting implementations.

The following consists of some abbreviated terms that are used in the tables below.

(SBF expands to SetBadFlag)

(kfs/KFs expands to KeyFrames) and (MP expands to MapPoint)

(LM expands to Local Mapping)

(LBA expands to Local Bundle Adjustment)

(No. expands to Number)

VANILLA – MH01

Table 10 Original-MH01-General Statistics

#Iterations	Total No. of kfs to have passed SBF	Current KF ID (Max)	Number of keyframes within map
1	142	445	304
2	140	442	303
3	141	438	298
4	140	443	304
5	142	448	307
6	149	454	306
7	142	450	309
8	149	452	304
9	147	448	302
10	145	447	303
	143.7	446.7	304

Table 11 Original-MH01-NetCount

#Iterations	Total number of MapPoints to have passed SBF	Number of MapPoints within map
1	18112	12144
2	18046	11993
3	17575	11971
4	17234	12154
5	17481	12198
6	17880	12252
7	17458	12242
8	18057	12216
9	17700	12130
10	17796	12098
	17733.9	12139.8

Table 12 Original-MH01-Local Mapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	6.8751	0.12931	33.656	245.55	29.621	314.66
2	7.0067	0.14964	34.191	248.81	30.131	319.09
3	6.8369	0.14527	33.433	248.72	28.742	316.68
4	6.8092	0.15084	33.261	241.75	29.107	310.02
5	6.8514	0.1312	32.048	246.85	29.572	314.25
6	6.6526	0.13112	31.994	241.8	27.896	307.32
7	6.7191	0.13053	32.052	239.05	28.87	305.69
8	6.7454	0.13379	31.704	240.55	27.099	305.11
9	6.6695	0.1311	32.384	239.31	28.469	305.83
10	6.7711	0.13129	32.361	239.04	28.581	305.73
	6.7937	0.136409	32.7084	243.143	28.8088	310.438

Table 13 Original-MH01-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	11.429	1.9982	5.8739	0.057652	21.283
2	11.56	2.0518	6.0017	0.060534	21.616
3	11.377	1.9832	5.7678	0.056727	21.059
4	11.535	1.9922	5.8708	0.057637	21.375
5	11.374	1.9582	5.9134	0.048207	21.15
6	11.377	1.9575	5.9459	0.049317	21.167
7	11.476	1.9379	5.8894	0.048433	21.21
8	11.409	1.9688	5.8038	0.04751	21.068
9	11.385	1.9594	5.8916	0.048042	21.117
10	11.448	1.9529	5.9301	0.050993	21.269
	11.437	1.97601	5.88884	0.0525052	21.2314

VANILLA – MH02

Table 14 Original-MH02-General Statistics

#Iterations	Total No. of kfs to have passed SBF	Current KF ID (Max)	Number of keyframes within map
1	97	373	277
2	115	381	267
3	106	376	271
4	110	374	265
5	103	359	257
6	110	368	259
7	114	379	266
8	113	386	274
9	101	366	266
10	102	369	268
	107.1	373.1	267

Table 15 Original-MH02-NetCount

#Iterations	Total number of MapPoints to have passed SBF	Number of MapPoints within map
1	16374	10906
2	15913	10801
3	16203	10692
4	16044	10677
5	15580	10277
6	15963	10515
7	15904	10903
8	16031	10901
9	16094	10676
10	15843	10712
	15994.9	10706

Table 16 Original-MH02-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	6.4949	0.14382	33.022	256.6	28.736	323.48
2	6.344	0.13726	32.983	246.02	26.978	310.2
3	6.4296	0.14639	33.034	248.3	27.018	313.54
4	6.5	0.1361	33.391	249.94	27.845	316.35
5	6.7582	0.14204	34.881	255.42	30.51	326.2
6	6.4004	0.13648	32.719	248.11	28.181	314.07
7	6.3546	0.14064	32.773	247.25	27.158	312.3
8	6.2559	0.13273	31.808	238.3	26.233	301.44
9	6.4176	0.13575	32.775	252.69	27.957	318.52
10	6.4645	0.14206	32.688	253.77	28.827	320.38
	6.44197	0.139327	33.0074	249.64	27.9443	315.648

Table 17 Original-MH02-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	11.177	1.9312	5.1604	0.051391	20.054
2	11.082	1.9045	4.9618	0.049309	19.695
3	11.302	1.934	5.0125	0.05214	20.062
4	11.193	1.9201	5.1507	0.056059	20.09
5	11.636	2.0664	5.5426	0.059168	21.147
6	11.252	1.9249	5.1157	0.052206	20.086
7	11.029	1.9193	4.9766	0.053714	19.697
8	11.184	1.8786	4.7814	0.048598	19.602
9	11.111	1.9325	5.1721	0.055251	20.007
10	11.075	1.9412	5.1394	0.051972	19.917
	11.2041	1.93527	5.10132	0.0529808	20.0357

VANILLA-MH03

Table 18 Original-MH03-General Statistics

#Iterations	Total No. of kfs to have passed SBF	Current KF ID (Max)	Number of keyframes within map
1	99	354	256
2	110	360	251
3	105	354	250
4	102	354	253
5	98	353	256
6	101	351	251
7	104	353	250
8	113	363	251
9	89	341	253
10	105	357	253
	102.6	354	252.4

Table 19 Original-MH03-NetCount

#Iterations	Total number of MapPoints to have passed SBF	Number of MapPoints within map
1	14269	9608
2	14880	9360
3	14523	9104
4	14555	9407
5	14541	9291
6	14380	9250
7	14498	9302
8	14619	9406
9	14134	9374
10	14706	9274
	14510.5	9337.6

Table 20 Original-MH03-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	6.2024	0.13637	27.699	214.15	26.261	273.1
2	6.2379	0.1419	28.546	214.19	25.869	273.66
3	6.057	0.13543	27.806	211.36	25.344	269.44
4	6.1682	0.13923	28.314	210.41	25.591	269.36
5	6.208	0.13094	27.887	213.03	26.336	272.24
6	6.0327	0.12843	27.809	206.14	25.862	264.65
7	6.2454	0.13436	28.359	214.25	26.568	274.27
8	6.1971	0.13447	28.433	214.19	26.936	274.64
9	6.038	0.13802	27.948	205.04	25.65	263.54
10	6.3073	0.13133	27.948	208	25.914	267.04
	6.1694	0.135048	28.0749	211.076	26.0331	270.194

Table 21 Original-MH03-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	11.185	1.9344	4.7243	0.051755	19.926
2	11.13	1.9248	4.5852	0.048744	19.678
3	11.198	1.9394	4.5658	0.051206	19.752
4	11.306	1.9612	4.7023	0.049703	20.044
5	11.175	1.8938	4.6266	0.050411	19.744
6	11.254	1.9022	4.5397	0.049311	19.741
7	11.252	1.9471	4.577	0.046607	19.825
8	11.252	1.9336	4.6801	0.051078	19.939
9	11.404	1.9468	4.6001	0.048014	20.015
10	11.226	1.9252	4.5695	0.051505	19.772
	11.2382	1.93085	4.61706	0.0498334	19.8436

Vanilla-MH04

Table 22 Original-MH04-General Statistics

#Iterations	Total No. of kfs to have passed SBF	Current KF ID (Max)	Number of keyframes within map
1	87	367	281
2	80	361	282
3	83	362	280
4	80	360	281
5	76	345	270
6	64	406	343
7	71	349	279
8	81	419	339
9	75	354	280
10	72	348	277
	76.9	367.1	291.2

Table 23 Original-MH04-NetCount

#Iterations	Total number of MapPoints to have passed SBF	Number of MapPoints within map
1	12597	11605
2	12425	11623
3	12513	11671
4	12453	11623
5	12052	11151
6	13675	15720
7	12218	11478
8	14082	14521
9	12465	11461
10	12457	11486
	12693.7	12233.9

Table 24 Original-MH04-Local-Mapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	5.1213	0.12782	22.436	140.18	20.697	187.75
2	5.1527	0.13063	22.472	145.08	20.623	192.6
3	5.0294	0.12723	22.252	142.14	20.743	189.45
4	4.9697	0.12264	22.353	141.64	20.154	188.4
5	5.2604	0.14076	24.356	149.91	22.059	200.8
6	4.968	0.13015	22.353	121.07	22.308	170.18
7	5.24887	0.13699	23.36091	143.23512	21.11351	192.21733
8	5.00303	0.14648	21.88037	113.99013	18.72405	159.15744
9	5.40495	0.14161	23.80956	145.34871	21.76804	195.58304
10	5.12352	0.13864	23.95864	146.19979	22.07715	196.59746
	5.128187	0.134295	22.923148	138.879375	21.026675	187.273527

Table 25 Original-MH04-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	11.111	2.119	4.398	0.05661	19.61
2	11.156	2.041	4.2395	0.052375	19.41
3	11.157	2.0409	4.2433	0.053279	19.427
4	11.058	1.9949	4.227	0.05066	19.246
5	11.462	2.1239	4.628	0.06315	20.279
6	11.688	2.0738	4.3211	0.058168	20.189
7	11.35424	2.01336	4.42048	0.0541	19.83561
8	11.75859	2.19188	3.82404	0.05869	19.75088
9	11.80902	2.194	4.54194	0.0606	20.64463
10	11.77218	2.159	4.43809	0.05687	20.42844
	11.432603	2.095174	4.328145	0.0564502	19.882056

Vanilla-MH05

Table 26 Original-MH05-General Statistics

#Iterations	Total No. of kfs to have passed SBF	Current KF ID (Max)	Number of keyframes within map
1	124	434	311
2	131	440	312
3	101	394	294
4	92	410	319
5	113	424	312
6	127	433	307
7	136	460	325
8	124	431	308
9	119	430	313
10	116	465	350
	118.3	432.1	315.1

Table 27 Original-MH05-NetCount

#Iterations	Total number of MapPoints to have passed SBF	Number of MapPoints within map
1	13930	12462
2	13994	12796
3	13289	11783
4	14059	13081
5	13894	12529
6	14163	12781
7	14017	13106
8	14016	12780
9	14164	12558
10	15117	14690
	14064.3	12856.6

Table 28 Original-MH05-Local Mapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	5.5124	0.12716	24.577	135.76	22.422	187.36
2	5.4208	0.12242	23.598	138.6	19.842	186.91
3	5.7393	0.13713	25.513	149.88	26.579	207.03
4	5.3896	0.13566	24.027	134.27	22.74	185.85
5	5.5701	0.16161	25.367	139.59	24.037	194.01
6	5.582	0.13168	24.637	144.94	21.332	195.52
7	5.2688	0.13395	23.824	130.43	20.121	179.12
8	5.4536	0.12859	23.951	141.81	20.951	191.59
9	5.4542	0.1319	23.768	141.75	20.72	191.12
10	5.2304	0.12739	23.281	128.99	19.921	176.57
	5.46212	0.133749	24.2543	138.602	21.8665	189.508

Table 29 Original-MH05-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	11.332	2.1212	4.1395	0.055449	19.948
2	10.85	2.1129	4.3296	0.056228	19.59
3	11.248	2.1387	4.562	0.059315	20.304
4	11.072	2.0653	4.2083	0.055244	19.651
5	11.04	2.0892	4.2216	0.05513	19.68
6	10.97	2.2035	4.6167	0.061052	20.227
7	10.993	2.0781	4.1099	0.063076	19.568
8	11.053	2.1788	4.3808	0.05892	19.975
9	10.824	2.0825	4.2049	0.056491	19.406
10	10.902	2.1122	4.2901	0.060654	19.64
	11.0284	2.11824	4.30634	0.0581559	19.7989

Vanilla-V101

Table 30 Original-V101-General Statistics

#Iterations	Total No. of kfs to have passed SBF	Current KF ID (Max)	Number of keyframes within map
1	138	376	239
2	145	380	236
3	144	377	234
4	144	380	237
5	149	379	231
6	135	370	236
7	139	375	237
8	133	370	238
9	137	370	234
10	136	368	233
	140	374.5	235.5

Table 31 Original-V101-NetCount

#Iterations	Total number of MapPoints to have passed SBF	Number of MapPoints within map
1	14997	9889
2	14879	9719
3	15249	9800
4	14796	9707
5	14818	9832
6	14824	9784
7	14956	9742
8	14782	9901
9	14653	9858
10	14588	9669
	14854.2	9790.1

Table 32 Original-V101-Local-Mapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	7.2572	0.14275	29.4	256.72	38.142	330.2
2	7.2843	0.15632	29.848	256.2	37.916	329.86
3	7.1968	0.15981	29.611	246.83	36.585	318.98
4	7.0819	0.1572	29.609	254.74	37.49	327.65
5	6.9757	0.14736	28.467	250.46	35.152	319.79
6	7.3758	0.15218	29.842	257.44	37.914	331.14
7	7.2116	0.15622	29.123	250.01	37.422	322.49
8	7.2642	0.16131	29.905	257.34	36.158	329.34
9	7.1926	0.15161	28.896	254.45	36.88	326.1
10	7.3178	0.16099	29.396	254.63	38.813	328.73
	7.21579	0.154575	29.4097	253.882	37.2472	326.428

Table 33 Original-V101-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	10.345	2.2403	5.577	0.061963	20.479
2	10.183	2.2025	5.6118	0.057764	20.318
3	10.325	2.278	5.6125	0.062321	20.519
4	10.331	2.179	5.5882	0.061652	20.449
5	10.127	2.1863	5.6164	0.057097	20.194
6	10.343	2.2368	5.6814	0.060263	20.597
7	10.284	2.2078	5.6179	0.059604	20.43
8	10.455	2.2477	5.61	0.061881	20.648
9	10.228	2.1797	5.6098	0.064154	20.388
10	10.36	2.2746	5.7577	0.059102	20.687
	10.2981	2.22327	5.62827	0.0605801	20.4709

Original V1_02

Table 34 Original-V102-General Statistics

#Iterations	Total No. of kfs to have passed SBF	Current KF ID (Max)	Number of keyframes within map
1	115	411	297
2	130	440	311
3	116	489	377
4	121	502	387
5	122	403	282
6	101	369	269
7	116	403	290
8	113	382	270
9	108	389	283
10	111	382	273
	115.3	417	303.9

Table 35 Original-V102-NetCount

#Iterations	Total number of MapPoints to have passed SBF	Number of MapPoints within map
1	14888	10743
2	15896	11647
3	16986	12617
4	17045	13586
5	14598	10418
6	13483	9926
7	13368	11272
8	14118	10168
9	14586	10374
10	13796	10211
	14876.4	11096.2

Table 36 Original-V102-Local Mapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	6.0688	0.16821	25.539	164.18	26.227	220.86
2	5.5822	0.13646	23.101	155.41	22.858	205.58
3	5.3155	0.13977	22.017	128.96	20.612	175.87
4	5.3208	0.14893	22.436	129.58	18.891	175.82
5	5.9398	0.13874	24.896	170.66	25.978	226.22
6	5.8791	0.14513	24.382	179.06	25.057	233.49
7	5.6205	0.13571	22.3	167.68	20.936	215.79
8	5.9473	0.14863	24.658	183.71	26.61	240.04
9	6.0671	0.16056	25.161	171.6	26.379	228.42
10	5.8601	0.14319	24.823	177.97	25.927	233.19
	5.76012	0.146533	23.9313	162.881	23.9475	215.528

Table 37 Original-V102-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	10.233	1.8705	5.2149	0.057535	19.257
2	10.061	1.978	5.1137	0.06344	19.117
3	10.24	1.9285	4.2654	0.058641	18.389
4	10.278	1.8967	4.3509	0.063869	18.474
5	9.968	1.8635	5.0731	0.053985	18.806
6	10.193	1.8458	5.0751	0.055394	19.008
7	10.006	1.9671	5.257	0.055383	19.18
8	10.186	1.8894	5.1073	0.054255	19.073
9	10.169	1.905	5.2223	0.057071	19.203
10	10.269	1.858	5.0649	0.056307	19.083
	10.1603	1.90025	4.97446	0.057588	18.959

Original V103

Table 38 Original-V103-General Statistics

#Iterations	Total No. of kfs to have passed SBF	Current KF ID (Max)	Number of keyframes within map
1	145	472	328
2	113	414	302
3	137	457	321
4	138	433	298
5	123	511	389
6	98	433	336
7	123	428	306
8	109	402	295
9	113	414	303
10	138	502	366
	123.7	446.6	324.4

Table 39 Original-V103-NetCount

#Iterations	Total number of MapPoints to have passed SBF	Number of MapPoints within map
1	18771	11855
2	18042	10821
3	18062	11442
4	18272	10665
5	20806	13168
6	18504	11753
7	18372	10937
8	17927	10681
9	18541	10853
10	19494	13478
	18679.1	11565.3

Table 40 Original-V103-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	4.7511	0.14284	19.572	114.64	16.859	153.57
2	5.2768	0.16355	22.019	135.48	18.172	180.42
3	4.8585	0.13842	19.808	114.3	15.558	153.32
4	4.8456	0.14929	19.994	114.86	15.48	152.77
5	4.7496	0.14723	20.17	106.96	15.631	146.46
6	5.0637	0.16358	21.467	129.75	17.238	173.04
7	5.2136	0.15575	21.913	133.18	18.5	177.95
8	5.4209	0.15817	22.149	133.83	18.604	179.45
9	5.3163	0.16399	22.085	138.2	18.652	183.66
10	4.4472	0.1481	18.327	102.99	15.793	140.35
	4.99433	0.153092	20.7504	122.419	17.0487	164.099

Table 41Original-V103-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	10.783	2.2153	3.5262	0.058506	19.213
2	10.851	2.2657	3.8268	0.062209	19.644
3	11.02	2.2959	3.3334	0.057085	19.272
4	11.638	2.4374	3.3946	0.064405	20.372
5	10.192	1.8764	3.764	0.065864	18.521
6	10.608	2.1653	3.5949	0.055564	19.033
7	10.429	2.0594	3.8585	0.057327	19.063
8	10.707	2.259	3.8695	0.057119	19.59
9	10.472	2.2068	3.7973	0.059943	19.167
10	10.805	2.1762	3.446	0.055805	19.22
	10.7505	2.19574	3.64112	0.0593827	19.3095

Original-V201

Table 42 Original-V201-General Statistics

#Iterations	Total No. of kfs to have passed SBF	Current KF ID (Max)	Number of keyframes within map
1	191	443	253
2	231	525	295
3	207	465	259
4	204	460	257
5	191	452	262
6	207	474	269
7	209	495	289
8	206	479	276
9	189	446	258
10	207	457	251
	204.2	469.6	266.9

Table 43 Original-V201-NetCount

#Iterations	Total number of MapPoints to have passed SBF	Number of MapPoints within map
1	12697	11220
2	14140	13456
3	12861	11686
4	12943	11616
5	13012	11702
6	12673	11548
7	13324	12452
8	12948	11460
9	12746	11572
10	13017	11411
	13036.1	11812.3

Table 44 Original-V201-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	6.1847	0.14652	22.91	145.64	25.416	199.58
2	5.5174	0.1279	19.968	113.73	17.97	155.85
3	6.0375	0.13734	22.442	137.93	22.894	188.8
4	6.0375	0.13403	22.311	138.54	23.441	189.81
5	6.17	0.13516	22.816	141.23	24.06	193.74
6	5.9309	0.14135	22.004	127.31	22.766	177.56
7	5.9014	0.13772	22.339	125.32	22.364	175.51
8	6.0706	0.1349	22.56	132.91	22.288	183.04
9	6.2139	0.14939	23.108	143.21	24.412	196.4
10	6.122	0.13085	22.724	141.3	24.35	193.96
	6.01859	0.137516	22.3182	134.712	22.9961	185.425

Table 45 Original-V201-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	10.666	2.4045	4.2316	0.057332	19.539
2	10.547	2.5159	3.4394	0.057096	18.639
3	10.501	2.413	4.0313	0.05618	19.085
4	10.484	2.4021	4.0369	0.06157	19.073
5	10.528	2.3875	4.0552	0.056813	19.16
6	10.793	2.3823	4.0508	0.066845	19.484
7	10.731	2.4594	3.9215	0.064269	19.301
8	10.774	2.392	3.8049	0.059267	19.147
9	10.646	2.4297	4.1275	0.059631	19.385
10	10.585	2.4269	4.1265	0.060515	19.321
	10.6255	2.42133	3.98256	0.0599518	19.2134

Table 46 Original-V202-General Statistics

#Iterations	Total No. of kfs to have passed SBF	Current KF ID (Max)	Number of keyframes within map
1	115	411	297
2	130	440	311
3	116	489	377
4	121	502	387
5	122	403	282
6	101	369	269
7	116	403	290
8	113	382	270
9	108	389	283
10	111	382	273
	115.3	417	303.9

Table 47 Original-V202-NetCount

#Iterations	Total number of MapPoints to have passed SBF	Number of MapPoints within map
1	14888	10743
2	15896	11647
3	16986	12617
4	17045	13586
5	14598	10418
6	13483	9926
7	13368	11272
8	14118	10168
9	14586	10374
10	13796	10211
	14876.4	11096.2

Table 48 Original-V202-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	6.0688	0.16821	25.539	164.18	26.227	220.86
2	5.5822	0.13646	23.101	155.41	22.858	205.58
3	5.3155	0.13977	22.017	128.96	20.612	175.87
4	5.3208	0.14893	22.436	129.58	18.891	175.82
5	5.9398	0.13874	24.896	170.66	25.978	226.22
6	5.8791	0.14513	24.382	179.06	25.057	233.49
7	5.6205	0.13571	22.3	167.68	20.936	215.79
8	5.9473	0.14863	24.658	183.71	26.61	240.04
9	6.0671	0.16056	25.161	171.6	26.379	228.42
10	5.8601	0.14319	24.823	177.97	25.927	233.19
	5.76012	0.146533	23.9313	162.881	23.9475	215.528

Table 49 Original-V202-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	10.233	1.8705	5.2149	0.057535	19.257
2	10.061	1.978	5.1137	0.06344	19.117
3	10.24	1.9285	4.2654	0.058641	18.389
4	10.278	1.8967	4.3509	0.063869	18.474
5	9.968	1.8635	5.0731	0.053985	18.806
6	10.193	1.8458	5.0751	0.055394	19.008
7	10.006	1.9671	5.257	0.055383	19.18
8	10.186	1.8894	5.1073	0.054255	19.073
9	10.169	1.905	5.2223	0.057071	19.203
10	10.269	1.858	5.0649	0.056307	19.083
	10.1603	1.90025	4.97446	0.057588	18.959

Original V203

Table 50 Original-V203-General Statistics

#Iterations	Total No. of kfs to have passed SBF	Current KF ID (Max)	Number of keyframes within map
1	138	484	348
2	143	465	323
3	136	458	323
4	134	524	392
5	134	479	346
6	126	546	421
7	155	542	389
8	154	476	323
	140	496.75	358.125

Table 51Original-V203-NetCount

#Iterations	Total number of MapPoints to have passed SBF	Number of MapPoints within map
1	18425	13130
2	17660	12537
3	17353	12445
4	18912	15135
5	18918	13886
6	19756	16336
7	19717	15750
8	17497	12725
	18529.75	13993

Table 52 Original-V203-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	5.1242	0.16201	20.003	109.83	16.471	150.61
2	5.0535	0.15894	19.076	108.15	16.165	147.34
3	4.7952	0.14888	17.677	85.831	13.425	120.87
4	5.07	0.1548	19.782	109.41	17.198	150.31
5	4.9624	0.16204	19.063	97.259	15.483	135.92
6	5.0596	0.15632	19.723	107.96	17.41	149.34
7	4.7049	0.14166	18.33	89.941	15.23	127.44
8	4.8115	0.14419	18.413	92.434	15.175	130.04
9	4.8913	0.14987	18.173	90.426	14.255	126.82
10	5.0002	0.13755	19.057	109.62	16.152	148.71
	4.94728	0.151626	18.9297	100.0861	15.6964	138.74

Table 53Original-V203-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	11.873	2.3179	3.9715	0.06711	20.712
2	11.959	2.3095	3.9375	0.059256	20.702
3	12.515	2.4499	3.6107	0.067482	21.308
4	11.754	2.3266	3.9105	0.064035	20.54
5	11.612	2.3448	3.9337	0.067083	20.546
6	11.552	2.3642	4.0495	0.062678	20.473
7	11.725	2.2061	3.6108	0.068554	20.278
8	11.221	2.3055	3.8139	0.069604	19.833
9	12.136	2.4672	3.7205	0.064615	20.991
10	12.198	2.4216	3.934	0.056931	21.022
	11.8545	2.35133	3.84926	0.0647348	20.6405

ORBSLAM3 Deletion Statistics with Thread Safe Reference Counting using Mutexes.

The following statistics are for mutex locked reference counted ORBSLAM3. It contains the same information as above along with deletion statistics. The times for measurements pertaining to function execution times are in milliseconds. The results are discussed at the end of this section.

The calculations for all quantities are averaged across 10 iterations of each sequence. The following tables represent the number of keyframes, and map points processed and marked for deletion. It also shows the number of deleted keyframes and map points along with related statistics. Execution times of the local mapping, loop closing, and the tracking thread are also tabulated below. These statistics were captured to evaluate both the reference counting implementations.

The following consists of some abbreviated terms that are used in the tables below.

(SBF expands to SetBadFlag)

(kfs/KFs expands to KeyFrames) and (MP expands to MapPoint)

(LM expands to Local Mapping)

(LBA expands to Local Bundle Adjustment)

(No. expands to Number)

RF-MH01

Table 54 RF-MH01-General Statistics

#Iterations	Total No. of kfs to pass SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	112	77	406	295	68.75%	27.58%
2	115	82	416	302	71.30%	27.64%
3	113	77	405	293	68.14%	27.90%
4	111	75	407	297	67.57%	27.27%
5	108	75	406	299	69.44%	26.60%
6	109	75	403	295	68.81%	27.05%
7	101	71	399	299	70.30%	25.31%
8	109	73	403	295	66.97%	27.05%
9	106	66	406	301	62.26%	26.11%
10	106	70	401	296	66.04%	26.43%
	109	74.1	405.2	297.2	67.96%	26.89%

Table 55 RF-MH01-MapPoint-Deletion Statistics

Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion	Percentage of deletion (wrt mappoints in maps)
15471	15237	11335	98.49%	134.42%
16121	15886	11692	98.54%	135.87%
15988	15759	11561	98.56%	136.31%
15943	15679	11478	98.34%	136.60%
16206	15967	11556	98.52%	138.17%
15602	15379	11376	98.57%	135.18%
15760	15513	11488	98.43%	135.03%
15823	15573	11683	98.42%	133.30%
14969	14751	10602	98.54%	139.13%
15449	15230	11452	98.58%	132.99%
15733.2	15497.4	11422.3	98.50%	135.70%

Table 56 RF-MH01-KeyFrame-Deletion Statistics

Total No. of kfs to have passed SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)	%Deletion wrt map
110	80	403	294	72.73%	27.30%	27.21
115	79	418	304	68.70%	27.51%	25.98
110	81	411	302	73.64%	26.76%	26.82
107	75	408	302	70.09%	26.23%	24.83
117	83	418	302	70.94%	27.99%	27.48
105	72	400	296	68.57%	26.25%	24.32
112	76	405	294	67.86%	27.65%	25.85
103	69	404	302	66.99%	25.50%	22.84
86	58	361	276	67.44%	23.82%	21.01
118	84	417	300	71.19%	28.30%	28
108.3	75.7	404.5	297.2	69.81%	26.73%	25.47

Table 57 RF-MH01-LocalMapping Statistics

KF Insertion	MP Culling	MP Culling	MP Creation	LBA	KF Culling	KF Culling	Total Local Mapping	Total Local Mapping
7.0074	8.5239	8.523	35.68	267	26.669	26.669	343.62	343.62
6.9441	8.3582	8.358	35.62	258.	27.528	27.528	335.88	335.88
7.0009	8.4246	8.424	36.4	265.71	28.16	28.16	344.37	344.37
7.0476	8.5316	8.531	35.66	264.2	27.316	27.316	341.43	341.43
7.0664	11.462	11.46	36.89	261.8	28.193	28.193	344.04	344.04
7.117	12.717	12.71	37.78	272.59	28.651	28.651	357.43	357.43
6.9941	13.175	13.1	37.7	268.38	27.817	27.817	352.76	352.76
7.2012	8.8628	8.862	38.12	265.18	29.368	29.368	347.35	347.35
7.224	12.573	12.57	37.78	264.9	28.465	28.465	349.58	349.58
	10.29				28.01		346.27	

Table 58 RF-MH01-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	11.815	2.1983	6.9519	0.0639	22.871
2	11.864	2.2035	6.8763	0.0604	22.82
3	11.65	2.2445	6.8626	0.0625	22.661
4	11.692	2.1879	6.8732	0.0614	22.631
5	12.015	2.2781	7.116	0.0688	23.347
6	12.261	2.3587	7.363	0.0706	24.012
7	12.127	2.3351	7.3708	0.0684	23.842
8	12.326	2.3602	7.2153	0.0708	23.929
9	12.095	2.3739	7.1626	0.0673	23.603
	11.98277778	2.282244444	7.087966667	0.06601111111	23.30177778

RF-MH02

Table 59 RF-MH02-General Statistics

#Iterations	Total No. of kfs to have passed SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	133	101	419	287	75.94%	35.19%	24.11%	31.74%
2	81	59	351	271	72.84%	21.77%	16.81%	23.08%
3	90	66	350	261	73.33%	25.28%	18.85%	25.71%
4	88	59	348	261	67.05%	22.61%	16.95%	25.28%
5	83	58	345	263	69.88%	22.05%	16.81%	24.06%
6	82	59	340	259	71.95%	22.78%	17.35%	24.12%
7	88	60	342	255	68.18%	23.53%	17.54%	25.73%
8	120	90	405	286	75%	31.47%	22.22%	29.63%
9	137	100	426	290	72.99%	34.48%	23.47%	32.16%
10	131	98	421	291	74.81%	33.68%	23.28%	31.12%
	103.3	75	374.7	272.4	72.20%	27.28%	19.74%	27.26%

Table 60 RF-MH02-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion	Percentage of deletion (wrt mappoints in maps)
1	13987	13786	9969	98.56%	138.29%
2	14380	14121	9947	98.20%	141.96%
3	14497	14260	10276	98.37%	138.77%
4	14270	14076	10408	98.64%	135.24%
5	14185	13925	10637	98.17%	130.91%
6	15043	14837	10587	98.63%	140.14%
7	14841	14609	10402	98.44%	140.44%
8	14156	13938	10446	98.46%	133.43%
9	14473	14216	10184	98.22%	139.59%
10	14306	14056	10612	98.25%	132.45%
	14413.8	14182.4	10346.8	98.39%	137.12%

Table 61 RF-MH02-KeyFrame-Deletion Statistics

#Iterations	Total No. of kfs to pass SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion	Percentage of kfs to pass SBF	%Deletion wrt map
1	89	61	339	251	68.54%	24.30%	17.99%	26.25%	24.302
2	98	68	354	257	69.39%	26.46%	19.21%	27.68%	26.45
3	88	64	350	263	72.73%	24.33%	18.29%	25.14%	24.33
4	89	59	357	269	66.29%	21.93%	16.53%	24.93%	21.93
5	86	57	363	278	66.28%	20.50%	15.70%	23.69%	20.50
6	93	68	364	272	73.12%	25%	18.68%	25.55%	25
7	92	60	359	268	65.22%	22.39%	16.71%	25.63%	22.38
8	91	61	363	273	67.03%	22.34%	16.80%	25.07%	22.34
9	93	60	352	260	64.52%	23.08%	17.05%	26.42%	23.0
10	90	61	363	274	67.78%	22.26%	16.80%	24.79%	22.26
	90.9	61.9	356.4	266.5	68.09%	23.26%	17.38%	25.52%	23.22

Table 62 RF-MH02-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	6.6639	8.739	36.622	274.81	24.74	349.94
2	6.4245	8.6598	35.53	268.13	23.458	340.6
3	6.5839	11.458	37.712	280.99	25.285	360.32
4	6.7975	9.1678	37.663	273.57	25.553	351.02
5	6.4939	8.5331	36.141	271.71	24.998	346.18
6	6.4896	8.4293	36.083	269.71	24.95	343.61
7	6.2428	8.3577	34.869	258.77	22.67	329.4
8	6.3203	8.5615	35.81	267.94	24.168	341.18
9	6.2548	8.2472	34.898	259.29	23.603	330.73
10	6.2372	8.25	34.296	261.2	22.573	331.07
	6.45084	8.84034	35.9624	268.612	24.1998	342.405

Table 63 RF-MH02-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	11.8	2.2288	5.8488	0.0574	21.661
2	12.069	2.2239	5.7325	0.0596	21.816
3	11.951	2.2352	5.7571	0.0579	21.721
4	11.95	2.2585	6.1657	0.0609	22.184
5	12.186	2.3003	5.9341	0.0691	22.256
6	11.988	2.2266	5.9568	0.0577	21.977
7	11.675	2.1193	5.6011	0.0507	21.116
8	11.654	2.1332	5.7519	0.0531	21.262
9	11.534	2.1166	5.5157	0.053	20.863
10	11.612	2.0838	5.4984	0.0532	20.888
	11.8419	2.19262	5.77621	0.05726	21.5744

RF-MH03

Table 64 RF-MH03-General Statistics

#Iterations	Total No. of kfs to pass SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (	Percentage of kfs passed SetBadFlag
1	115	47	390	276	40.87%	17.03%	12.05%	29.49%
2	108	48	394	287	44.44%	16.72%	12.18%	27.41%
3	126	54	403	278	42.86%	19.42%	13.40%	31.27%
4	116	48	392	277	41.38%	17.33%	12.24%	29.59%
5	116	50	390	275	43.10%	18.18%	12.82%	29.74%
6	114	53	391	278	46.49%	19.06%	13.55%	29.16%
7	121	53	396	276	43.80%	19.20%	13.38%	30.56%
8	120	53	398	279	44.17%	19.00%	13.32%	30.15%
9	116	39	389	274	33.62%	14.23%	10.03%	29.82%
10	119	47	394	276	39.50%	17.03%	11.93%	30.20%
	117.1	49.2	393.7	277.6	42.02%	17.72%	12.49%	29.74%

Table 65 RF-MH03-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion	Percentage of deletion (wrt mappoints in maps)
1	13038	12667	9168	97.15%	138.17%
2	12867	12459	9260	96.83%	134.55%
3	13322	12938	9019	97.12%	143.45%
4	12822	12444	9254	97.05%	134.47%
5	12952	12599	9042	97.27%	139.34%
6	12924	12519	9202	96.87%	136.05%
7	13031	12673	9192	97.25%	137.87%
8	13302	12906	9281	97.02%	139.06%
9	13070	12687	9062	97.07%	140.00%
10	12856	12454	9052	96.87%	137.58%
	13018.4	12634.6	9153.2	97.05%	138.05%

Table 66 RF-MH03-KeyFrame-Deletion Statistics

#Iterations	Total No. of kfs to have passed SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	91	33	342	252	36.26%	13.10%	9.65%	26.61%
2	89	36	338	250	40.45%	14.40%	10.65%	26.33%
3	96	36	343	248	37.50%	14.52%	10.50%	27.99%
4	80	32	331	252	40%	12.70%	9.67%	24.17%
5	89	30	335	247	33.71%	12.15%	8.96%	26.57%
6	81	27	330	250	33.33%	10.80%	8.18%	24.55%
7	87	33	335	249	37.93%	13.25%	9.85%	25.97%
8	94	31	346	253	32.98%	12.25%	8.96%	27.17%
9	84	28	333	250	33.33%	11.20%	8.41%	25.23%
10	86	32	333	248	37.21%	12.90%	9.61%	25.83%
	87.7	31.8	336.6	249.9	36.27%	12.73%	9.44%	26.04%

Table 67 RF-MH03-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	6.176	8.753	30.255	229.84	25.112	298.71
2	6.33	9.156	31.366	236.82	26.215	308.33
3	6.425	9.252	32.066	244.06	25.439	315.62
4	6.388	9.132	31.699	237.4	25.335	308.41
5	6.193	9.157	30.551	229.16	24.14	297.75
6	6.155	9.042	30.695	230.96	24.651	300.03
7	6.247	8.985	30.769	238.09	25.267	307.84
8	6.158	8.79	30.306	229.15	24.54	297.48
9	6.15	8.921	30.474	233.19	24.647	301.89
10	6.038	8.967	30.157	224.16	23.923	291.84
	6.226	9.0155	30.8338	233.283	24.9269	302.79

Table 68 RF-MH03 Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	11.401	2.0901	5.3736	0.0513	20.873
2	12.045	2.2602	5.9458	0.065	22.37
3	12.076	2.299	5.8548	0.0661	22.408
4	12.039	2.294	5.7771	0.0569	22.269
5	11.635	2.1834	5.469	0.0574	21.347
6	11.84	2.203	5.459	0.0541	21.565
7	11.661	2.1505	5.4058	0.055	21.279
8	11.68	2.1629	5.5249	0.0528	21.378
9	11.712	2.1404	5.4836	0.0515	21.379
10	11.768	2.1686	5.3936	0.0543	21.372
	11.7857	2.19521	5.56872	0.05644	21.624

RF-MH04

Table 69 RF-MH04-General Statistics

#Iterations	Total No. of kfs to have passed SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	91	77	399	309	84.62%	24.92%	19.30%	22.81%
2	91	76	396	306	83.52%	24.84%	19.19%	22.98%
3	92	74	393	302	80.43%	24.50%	18.83%	23.41%
4	110	89	413	304	80.91%	29.28%	21.55%	26.63%
5	99	84	403	305	84.85%	27.54%	20.84%	24.57%
6	104	84	417	314	80.77%	26.75%	20.14%	24.94%
7	99	80	407	309	80.81%	25.89%	19.66%	24.32%
8	94	79	396	303	84.04%	26.07%	19.95%	23.74%
9	93	76	423	331	81.72%	22.96%	17.97%	21.99%
10	98	76	397	300	77.55%	25.33%	19.14%	24.69%
	97.1	79.5	404.4	308.3	81.92%	25.81%	19.66%	24.01%

Table 70 RF-MH04-MapPoint-Deletion-Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion	Percentage of deletion (wrt mappoints in maps)
1	11241	11010	11352	97.95%	96.99%
2	11130	10910	11295	98.02%	96.59%
3	11144	10893	11378	97.75%	95.74%
4	11137	10896	11183	97.84%	97.43%
5	11099	10870	11205	97.94%	97.01%
6	11283	11013	11389	97.61%	96.70%
7	11010	10735	11486	97.50%	93.46%
8	11043	10812	11370	97.91%	95.09%
9	10959	10764	11156	98.22%	96.49%
10	11081	10827	11341	97.71%	95.47%
	11112.7	10873	11315.5	97.84%	96.10%

Table 71RF-MH04-KeyFrame-Deletion Statistics

#Iterations	Total No. of kfs to have passed SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	65	51	345	281	78.46%	18.15%	14.78%	18.84%
2	57	46	334	278	80.70%	16.55%	13.77%	17.07%
3	60	47	338	279	78.33%	16.85%	13.91%	17.75%
4	65	50	340	276	76.92%	18.12%	14.71%	19.12%
5	64	52	338	275	81.25%	18.91%	15.38%	18.93%
6	60	47	339	280	78.33%	16.79%	13.86%	17.70%
7	59	48	336	278	81.36%	17.27%	14.29%	17.56%
8	67	53	342	276	79.10%	19.20%	15.50%	19.59%
9	53	40	328	276	75.47%	14.49%	12.20%	16.16%
10	60	50	336	277	83.33%	18.05%	14.88%	17.86%
	61	48.4	337.6	277.6	79.33%	17.44%	14.33%	18.06%

Table 72 RF-MH04-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	4.9852	9.2633	24.008	155.28	18.964	211.52
2	5.0881	9.2505	24.383	154.24	19.605	211.56
3	4.9308	9.7733	24.004	138.35	18.918	195.14
4	5.1766	9.8959	25.004	157.34	19.104	215.52
5	5.0618	9.0668	24.903	158.68	19.531	216.22
6	4.9853	9.2586	24.375	157.5	18.897	214.02
7	5.1972	9.4324	25.001	160.28	19.886	218.76
8	4.9031	9.036	22.8	144.75	18.375	198.99
9	4.9258	9.2048	23.373	142.56	18.531	197.75
10	5.0553	9.0813	24.08	154.99	19.431	211.64
	5.03092	9.32629	24.1931	152.397	19.1242	209.112

Table 73 RF-MH04-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	11.737	2.3412	5.1011	0.0558	21.116
2	11.783	2.291	5.1256	0.0562	21.175
3	12.213	2.2861	4.8277	0.0602	21.365
4	11.89	2.4307	5.2411	0.0637	21.603
5	11.654	2.3084	5.102	0.0564	21.037
6	11.628	2.277	5.0132	0.0576	20.897
7	11.879	2.3469	5.1567	0.0528	21.362
8	11.597	2.2122	4.9072	0.0602	20.67
9	11.675	2.1877	4.8506	0.0593	20.688
10	11.865	2.2627	4.9088	0.053	20.985
	11.7921	2.29439	5.0234	0.05752	21.0898

RF-MH05

Table 74 RF-MH05-General Statistics

#Iterations	Total No. of kfs to have passed SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	156	124	491	336	79.49%	36.90%	25.25%	31.77%
2	135	112	460	326	82.96%	34.36%	24.35%	29.35%
3	145	125	481	337	86.21%	37.09%	25.99%	30.15%
4	147	121	471	325	82.31%	37.23%	25.69%	31.21%
5	142	116	466	325	81.69%	35.69%	24.89%	30.47%
6	160	132	498	339	82.50%	38.94%	26.51%	32.13%
7	143	114	461	319	79.72%	35.74%	24.73%	31.02%
8	136	130	551	416	95.59%	31.25%	23.59%	24.68%
9	144	121	465	322	84.03%	37.58%	26.02%	30.97%
10	148	120	490	343	81.08%	34.99%	24.49%	30.20%

Table 75 RF-MH05-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion	Percentage of deletion (wrt mappoints in maps)
1	12079	11823	11727	97.88%	100.82%
2	13006	12802	12844	98.43%	99.67%
3	11968	11761	11354	98.27%	103.58%
4	12723	12497	12786	98.22%	97.74%
5	12702	12523	12845	98.59%	97.49%
6	11831	11647	11212	98.44%	103.88%
7	12703	12474	12641	98.20%	98.68%
8	12257	12060	12675	98.39%	95.15%
9	12427	12196	12237	98.14%	99.66%
10	12571	12320	12195	98.00%	101.03%
	12426.7	12210.3	12251.6	98.26%	99.77%

Table 76 RF-MH05-KeyFrame-Deletion Statistics

#Iterations	Total No. of kfs to pass SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	97	81	391	295	83.51%	27.46%	20.72%	24.81%
2	83	74	410	328	89.16%	22.56%	18.05%	20.24%
3	80	63	368	289	78.75%	21.80%	17.12%	21.74%
4	112	89	424	313	79.46%	28.43%	20.99%	26.42%
5	97	81	423	327	83.51%	24.77%	19.15%	22.93%
6	87	74	365	279	85.06%	26.52%	20.27%	23.84%
7	103	79	410	308	76.70%	25.65%	19.27%	25.12%
8	102	81	409	308	79.41%	26.30%	19.80%	24.94%
9	97	78	398	302	80.41%	25.83%	19.60%	24.37%
10	94	73	393	300	77.66%	24.33%	18.58%	23.92%
	95.2	77.3	399.1	304.9	81.36%	25.37%	19.35%	23.83%

Table 77 RF-MH05-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	5.42	8.726	25.42	157.2	19.02	215
2	5.396	8.638	25.06	148.3	18.45	204.5
3	5.508	8.726	26.07	159.1	22.47	221
4	4.878	8.822	23.61	122.9	20.67	180.4
5	5.503	8.541	25.29	153.7	19.03	211.3
6	5.527	8.881	26.09	159.8	22.22	221.6
7	5.651	8.664	26.23	165.2	22.71	227.5
8	5.383	8.404	25.81	155.9	21.87	216.5
9	5.554	8.659	25.75	158.8	22.14	220
10	5.426	8.84	25.57	155	19.21	213.3
	5.4246	8.6901	25.49	153.59	20.779	213.11

Table 78 RF-MH05-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	11.154	2.3005	5.1039	0.0645	20.845
2	11.18	2.2707	5.0923	0.0635	20.815
3	11.118	2.2157	5.0905	0.0585	20.673
4	10.989	2.2206	5.0698	0.0672	20.602
5	11.033	2.2514	5.1577	0.0568	20.724
6	11.123	2.1976	5.0523	0.0542	20.594
7	11.053	2.2025	5.0954	0.0588	20.569
8	11.033	2.1999	5.0618	0.0528	20.489
9	10.899	2.187	5.0453	0.0556	20.34
10	11.242	2.3337	5.3188	0.0723	21.216
	11.0824	2.23796	5.10878	0.06042	20.6867

RF-V101

Table 79 RF-V101 General Statistics

#Iterations	Total No. of kfs to have passed SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	183	119	430	248	65.03%	47.98%	27.67%	42.56%
2	177	119	422	246	67.23%	48.37%	28.20%	41.94%
3	167	108	417	251	64.67%	43.03%	25.90%	40.05%
4	174	107	422	249	61.49%	42.97%	25.36%	41.23%
5	169	110	422	254	65.09%	43.31%	26.07%	40.05%
6	182	123	432	251	67.58%	49.00%	28.47%	42.13%
7	174	116	424	251	66.67%	46.22%	27.36%	41.04%
8	168	110	418	251	65.48%	43.82%	26.32%	40.19%
9	164	112	412	249	68.29%	44.98%	27.18%	39.81%
10	171	120	426	256	70.18%	46.88%	28.17%	40.14%

Table 80 RF-V101-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion	Percentage of deletion (wrt mappoints in maps)
1	13225	12856	9658	97.21%	133.11%
2	13304	12932	9668	97.20%	133.76%
3	13167	12825	9526	97.40%	134.63%
4	12959	12563	9323	96.94%	134.75%
5	13145	12750	9396	97.00%	135.70%
6	12918	12550	9445	97.15%	132.87%
7	12996	12621	9321	97.11%	135.40%
8	13023	12673	9360	97.31%	135.40%
9	12856	12453	9379	96.87%	132.78%
10	13229	12883	9439	97.38%	136.49%
	13082.2	12710.6	9451.5	97.16%	134.49%

Table 81 RF-V101-KeyFrame-Deletion Statistics

#Iterations	Total No. of kfs to pass SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	120	75	353	234	62.50%	32.05%	21.25%	33.99%
2	117	72	352	236	61.54%	30.51%	20.45%	33.24%
3	121	73	351	231	60.33%	31.60%	20.80%	34.47%
4	113	68	339	227	60.18%	29.96%	20.06%	33.33%
5	109	65	332	224	59.63%	29.02%	19.58%	32.83%
6	116	70	345	230	60.34%	30.43%	20.29%	33.62%
7	106	60	331	226	56.60%	26.55%	18.13%	32.02%
8	108	62	337	230	57.41%	26.96%	18.40%	32.05%
9	106	65	331	226	61.32%	28.76%	19.64%	32.02%
10	116	68	346	231	58.62%	29.44%	19.65%	33.53%
	113.2	67.8	341.7	229.5	59.85%	29.53%	19.82%	33.11%

Table 82 RF-V101-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	7.1838	10.933	31.764	273.46	34.87	356.55
2	7.2691	10.906	31.748	279.2	34.787	362.24
3	7.3118	10.861	32.097	275.09	34.48	358.18
4	7.2334	11.063	31.905	276.45	35.053	360.05
5	7.1629	10.616	31.392	270.88	34.81	353.25
6	7.32	11.169	32.107	283.37	34.098	366.34
7	7.2175	10.921	31.643	273.84	34.78	356.77
8	7.3012	11.127	32.369	280.74	34.739	364.56
9	7.1566	11.009	31.57	276.43	34.659	359.11
10	7.2556	11.017	31.615	270.6	35.297	354.14
	7.24119	10.9622	31.821	276.006	34.7573	359.119

Table 83 RF-V101-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	10.653	2.4017	6.6388	0.0591	21.927
2	10.586	2.4002	6.6863	0.0616	21.911
3	10.612	2.435	6.7033	0.0625	21.969
4	10.535	2.4132	6.7321	0.0621	21.917
5	10.651	2.4074	6.631	0.0633	21.943
6	10.651	2.4074	6.631	0.0633	21.943
7	10.646	2.4295	6.6632	0.06	22.006
8	10.554	2.4406	6.7672	0.0647	22.016
9	10.578	2.4032	6.7038	0.063	21.895
10	10.575	2.4074	6.6006	0.057	21.81
	10.6041	2.41456	6.67573	0.06166	21.9337

RF-V102

Table 84 RF-V102-General Statistics

#Iterations	Total No. of kfs to have passed SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	91	51	362	272	56.04%	18.75%	14.09%	25.14%
2	82	53	364	283	64.63%	18.73%	14.56%	22.53%
3	93	60	379	287	64.52%	20.91%	15.83%	24.54%
4	80	49	348	269	61.25%	18.22%	14.08%	22.99%
5	57	50	307	252	87.72%	19.84%	16.29%	18.57%
6	90	51	356	268	56.67%	19.03%	14.33%	25.28%
7	87	55	357	271	63.22%	20.30%	15.41%	24.37%
8	82	48	343	262	58.54%	18.32%	13.99%	23.91%
9	85	60	378	294	70.59%	20.41%	15.87%	22.49%
10	87	55	380	294	63.22%	18.71%	14.47%	22.89%

Table 85 RF-V102-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion	Percentage of deletion (wrt mappoints in maps)
1	12435	12138	9539	97.61%	127.25%
2	11496	11174	10764	97.20%	103.81%
3	11667	11403	10655	97.74%	107.02%
4	12661	12077	11330	95.39%	106.59%
5	12562	11868	11106	94.48%	106.86%
6	13034	12730	10290	97.67%	123.71%
7	13391	12664	11389	94.57%	111.20%
8	11950	11624	10692	97.27%	108.72%
9	12622	12314	10018	97.56%	122.92%
10	12430	12119	10859	97.50%	111.60%
	12424.8	12011.1	10664.2	96.70%	112.97%

Table 86 RF-V102-KeyFrame-Deletion Statistics

#Iterations	Total No. of kfs to have passed SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	90	48	355	266	53.33%	18.05%	13.52%	25.35%
2	86	53	363	279	61.63%	19.00%	14.60%	23.69%
3	76	44	355	280	57.89%	15.71%	12.39%	21.41%
4	78	51	371	294	65.38%	17.35%	13.75%	21.02%
5	88	55	373	287	62.50%	19.16%	14.75%	23.59%
6	103	62	389	287	60.19%	21.60%	15.94%	26.48%
7	82	48	370	289	58.54%	16.61%	12.97%	22.16%
8	99	55	381	284	55.56%	19.37%	14.44%	25.98%
9	96	53	368	273	55.21%	19.41%	14.40%	26.09%
10	84	48	383	300	57.14%	16%	12.53%	21.93%
	88.2	51.7	370.8	283.9	58.74%	18.23%	13.93%	23.77%

Table 87 RF-V102-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	5.726	10.27	25.23	197.8	23.22	260.5
2	5.715	10.35	25.34	193.2	23.97	257.4
3	5.877	9.962	25.44	180.3	24.52	245.4
4	5.361	12.98	24.05	183.6	20.49	243.9
5	5.178	11.14	23.39	117.1	19.27	175
6	5.701	10.44	25.06	193.2	21.23	254.6
7	5.91	10.67	26.47	202.2	23.59	267.6
8	5.434	12.17	24.26	176.7	21.39	238.1
9	5.442	11.98	24.06	181	21.15	241.1
10	5.958	10.27	26.55	186.4	24.89	253
	5.6302	11.0232	24.985	181.15	22.372	243.66

Table 88 RF-V102-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	10.26	2.021	5.74	0.054	19.8
2	10.34	1.919	5.849	0.056	19.93
3	10.21	2.027	5.917	0.056	19.95
4	10.7	2.187	5.716	0.06	20.55
5	10.7	2.178	5.299	0.075	20.25
6	10.73	2.259	6.469	0.062	21.43
7	10.53	2.161	5.815	0.057	20.4
8	10.67	2.146	5.928	0.054	20.63
9	10.68	2.107	6	0.065	20.74
10	10.44	2.045	6.133	0.052	20.45
	10.526	2.105	5.8866	0.0591	20.413

RF-V103

Table 89 RF-V103-General Statistics

#Iterations	Total No. of kfs to have passed SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	89	56	378	290	62.92%	19.31%	14.81%	23.54%
2	88	62	467	382	70.45%	16.23%	13.28%	18.84%
3	125	87	453	48	69.60%	181.25%	19.21%	27.59%
4	117	71	408	292	60.68%	24.32%	17.40%	28.68%
5	125	103	561	437	82.40%	23.57%	18.36%	22.28%
6	124	90	439	289	72.58%	31.14%	20.50%	28.25%
7	79	44	378	300	55.70%	14.67%	11.64%	20.90%
8	100	84	499	400	84%	21%	16.83%	20.04%
9	120	93	466	347	77.50%	26.80%	19.96%	25.75%
10	93	52	386	294	55.91%	17.69%	13.47%	24.09%

Table 90 RF-V103-MapPoint Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion	Percentage of deletion (wrt mappoints in maps)
1	16382	15716	12492	95.93%	125.81%
2	15690	14970	10841	95.41%	138.09%
3	16997	16382	15172	96.38%	107.98%
4	15785	15126	11233	95.83%	134.66%
5	16698	15957	1543	95.56%	1034.15%
6	16313	15718	13615	96.35%	115.45%
7	17036	16158	13942	94.85%	115.89%
8	16072	14806	13040	92.12%	113.54%
9	15187	14763	10918	97.21%	135.22%
10	15451	15010	10488	97.15%	143.12%
	16161.1	15460.6	11328.4	95.68%	216.39%

Table 91RF-V103-KeyFrame-Deletion Statistics

#Iterations	Total No. of kfs to have passed SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	100	66	459	360	66%	18.33%	14.38%	21.79%
2	127	87	464	338	68.50%	25.74%	18.75%	27.37%
3	113	86	532	420	76.11%	20.48%	16.17%	21.24%
4	97	57	415	320	58.76%	17.81%	13.73%	23.37%
5	105	80	508	45	76.19%	177.78%	15.75%	20.67%
6	91	72	476	387	79.12%	18.60%	15.13%	19.12%
7	130	99	531	404	76.15%	24.50%	18.64%	24.48%
8	78	60	478	401	76.92%	14.96%	12.55%	16.32%
9	88	58	412	325	65.91%	17.85%	14.08%	21.36%
10	80	38	367	289	47.50%	13.15%	10.35%	21.80%
	100.9	70.3	464.2	328.9	69%	34.92%	14.95%	21.75%

Table 92 RF-V103-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	4.329	12.94	18.95	108	11.54	153.6
2	4.357	11.73	19.29	107.5	13.38	154.3
3	4.175	11.29	18.95	109.9	13.44	156.5
4	4.454	13.25	19.94	109.8	15.3	161.2
5	4.408	11.1	20.55	119.4	14.06	168.2
6	4.211	10.5	18.72	91.77	11.67	136.3
7	4.446	13.08	20.19	121.1	13.36	170.4
8	4.29	12.88	19.57	111.8	12.16	159.2
9	4.284	10.88	19.18	97.66	13.44	144.6
10	4.188	12.16	18.46	99.34	11	143.4
	4.3142	11.981	19.38	107.627	12.935	154.77

Table 93 RF-V103-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	11.75	2.461	3.563	0.047	20.5
2	11.23	2.391	3.78	0.053	20.04
3	10.97	2.329	3.864	0.05	19.81
4	10.5	2.297	4.399	0.062	19.91
5	10.73	2.1	4.031	0.052	19.47
6	10.06	2.098	3.698	0.059	18.52
7	11.02	2.281	4.115	0.052	20.12
8	11.11	2.368	3.766	0.053	19.89
9	9.974	2.113	3.901	0.057	18.63
10	10.7	2.24	3.499	0.054	19.08
	10.8044	2.2678	3.8616	0.0539	19.597

RF-V201

Table 94 RF-V201-General Statistics

#Iterations	Total No. of kfs to have passed SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	156	147	413	258	94.23%	56.98%	35.59%	37.77%
2	152	144	413	263	94.74%	54.75%	34.87%	36.80%
3	148	139	392	245	93.92%	56.73%	35.46%	37.76%
4	143	136	387	245	95.10%	55.51%	35.14%	36.95%
5	164	153	419	257	93.29%	59.53%	36.52%	39.14%
6	144	134	375	232	93.06%	57.76%	35.73%	38.40%
7	145	132	382	238	91.03%	55.46%	34.55%	37.96%
8	162	152	426	266	93.83%	57.14%	35.68%	38.03%
9	145	135	385	241	93.10%	56.02%	35.06%	37.66%
10	150	140	408	260	93.33%	53.85%	34.31%	36.76%

Table 95 RF-V201-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion	Percentage of deletion (wrt mappoints in maps)
1	11302	11010	10798	97.42%	101.96%
2	10815	10523	10836	97.30%	97.11%
3	11074	10771	11105	97.26%	96.99%
4	10679	10352	10730	96.94%	96.48%
5	10746	10465	10986	97.39%	95.26%
6	11509	11179	10792	97.13%	103.59%
7	10889	10588	10837	97.24%	97.70%
8	11189	10837	11172	96.85%	97.00%
9	10761	10451	10641	97.12%	98.21%
10	11131	10808	10300	97.10%	104.93%
	11009.5	10698.4	10819.7	97.17%	98.92%

Table 96 94 RF-V201-KeyFrame-Deletion Statistics

#Iterations	Total No. of kfs to have passed SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	143	134	404	264	93.71%	50.76%	33.17%	35.40%
2	138	128	382	245	92.75%	52.24%	33.51%	36.13%
3	144	134	407	265	93.06%	50.57%	32.92%	35.38%
4	141	132	383	243	93.62%	54.32%	34.46%	36.81%
5	127	117	377	251	92.13%	46.61%	31.03%	33.69%
6	153	139	407	256	90.85%	54.30%	34.15%	37.59%
7	139	130	382	244	93.53%	53.28%	34.03%	36.39%
8	149	138	414	267	92.62%	51.69%	33.33%	35.99%
9	135	126	377	243	93.33%	51.85%	33.42%	35.81%
10	150	132	408	260	88%	50.77%	32.35%	36.76%
	141.9	131	394.1	253.8	92.36%	51.64%	33.24%	35.99%

Table 97 RF-V201-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	6.229	10.79	25.64	161.4	23.62	226.8
2	6.258	10.85	25.28	157.9	22.53	222
3	6.255	11.15	25.37	154.9	23.65	220.5
4	5.899	10.88	23.89	142.3	20.36	202.7
5	5.879	10.87	24.15	142.5	21.06	203.8
6	5.897	10.61	23.5	142.8	21.24	203.4
7	6.108	10.94	25.28	154.1	23.25	218.9
8	5.812	11	23.99	144.3	20.07	203.7
9	5.863	10.64	23.85	143.7	20.02	203.4
10	5.952	10.67	24.21	145.6	20.87	206.2
	6.0152	10.84	24.516	148.95	21.667	211.14

Table 98 RF-V201-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	10.91	2.598	5.006	0.06	20.59
2	10.95	2.637	4.957	0.058	20.62
3	10.89	2.591	5.044	0.063	20.63
4	10.87	2.553	4.562	0.058	20.05
5	11.01	2.572	4.742	0.063	20.45
6	10.96	2.527	4.702	0.058	20.29
7	11.01	2.62	4.889	0.058	20.65
8	10.9	2.561	4.75	0.058	20.3
9	11.02	2.568	4.602	0.058	20.34
10	10.88	2.578	4.604	0.057	20.16
	10.94	2.5805	4.7858	0.0591	20.408

RF-V202

Table 99 RF-V202-General Statistics

#Iterations	Total No. of kfs to have passed SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	91	51	362	272	56.04%	18.75%	14.09%	25.14%
2	82	53	364	283	64.63%	18.73%	14.56%	22.53%
3	93	60	379	287	64.52%	20.91%	15.83%	24.54%
4	80	49	348	269	61.25%	18.22%	14.08%	22.99%
5	57	50	307	252	87.72%	19.84%	16.29%	18.57%
6	90	51	356	268	56.67%	19.03%	14.33%	25.28%
7	87	55	357	271	63.22%	20.30%	15.41%	24.37%
8	82	48	343	262	58.54%	18.32%	13.99%	23.91%
9	85	60	378	294	70.59%	20.41%	15.87%	22.49%
10	87	55	380	294	63.22%	18.71%	14.47%	22.89%

Table 100 RF-V202-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion	Percentage of deletion (wrt mappoints in maps)
1	12435	12138	9539	97.61%	127.25%
2	11496	11174	10764	97.20%	103.81%
3	11667	11403	10655	97.74%	107.02%
4	12661	12077	11330	95.39%	106.59%
5	12562	11868	11106	94.48%	106.86%
6	13034	12730	10290	97.67%	123.71%
7	13391	12664	11389	94.57%	111.20%
8	11950	11624	10692	97.27%	108.72%
9	12622	12314	10018	97.56%	122.92%
10	12430	12119	10859	97.50%	111.60%
	12424.8	12011.1	10664.2	96.70%	112.97%

Table 101RF-V202-KeyFrame-Deletion Statistics

#Iterations	Total No. of kfs to have passed SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	90	48	355	266	53.33%	18.05%	13.52%	25.35%
2	86	53	363	279	61.63%	19.00%	14.60%	23.69%
3	76	44	355	280	57.89%	15.71%	12.39%	21.41%
4	78	51	371	294	65.38%	17.35%	13.75%	21.02%
5	88	55	373	287	62.50%	19.16%	14.75%	23.59%
6	103	62	389	287	60.19%	21.60%	15.94%	26.48%
7	82	48	370	289	58.54%	16.61%	12.97%	22.16%
8	99	55	381	284	55.56%	19.37%	14.44%	25.98%
9	96	53	368	273	55.21%	19.41%	14.40%	26.09%
10	84	48	383	300	57.14%	16%	12.53%	21.93%
	88.2	51.7	370.8	283.9	58.74%	18.23%	13.93%	23.77%

Table 102 RF-V202-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	5.726	10.27	25.23	197.8	23.22	260.5
2	5.715	10.35	25.34	193.2	23.97	257.4
3	5.877	9.962	25.44	180.3	24.52	245.4
4	5.361	12.98	24.05	183.6	20.49	243.9
5	5.178	11.14	23.39	117.1	19.27	175
6	5.701	10.44	25.06	193.2	21.23	254.6
7	5.91	10.67	26.47	202.2	23.59	267.6
8	5.434	12.17	24.26	176.7	21.39	238.1
9	5.442	11.98	24.06	181	21.15	241.1
10	5.958	10.27	26.55	186.4	24.89	253
	5.6302	11.0232	24.985	181.15	22.372	243.66

Table 103 RF-V202-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	10.26	2.021	5.74	0.054	19.8
2	10.34	1.919	5.849	0.056	19.93
3	10.21	2.027	5.917	0.056	19.95
4	10.7	2.187	5.716	0.06	20.55
5	10.7	2.178	5.299	0.075	20.25
6	10.73	2.259	6.469	0.062	21.43
7	10.53	2.161	5.815	0.057	20.4
8	10.67	2.146	5.928	0.054	20.63
9	10.68	2.107	6	0.065	20.74
10	10.44	2.045	6.133	0.052	20.45
	10.526	2.105	5.8866	0.0591	20.413

RF-V203

Table 104 RF-V203-General Statistics

#Iterations	Total No. of kfs to have passed SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	104	62	458	356	59.62%	17.42%	13.54%	22.71%
2	110	65	490	384	59.09%	16.93%	13.27%	22.45%
3	63	53	271	209	84.13%	25.36%	19.56%	23.25%
4	107	59	474	368	55.14%	16.03%	12.45%	22.57%
5	111	66	460	351	59.46%	18.80%	14.35%	24.13%
6	105	67	432	329	63.81%	20.36%	15.51%	24.31%
7	98	64	477	381	65.31%	16.80%	13.42%	20.55%
8	104	66	490	388	63.46%	17.01%	13.47%	21.22%
9	115	70	513	401	60.87%	17.46%	13.65%	22.42%
10	105	76	464	360	72.38%	21.11%	16.38%	22.63%

Table 105 RF-V203-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion	Percentage of deletion (wrt mappoints in maps)
1	14622	14133	12203	96.66%	115.82%
2	14696	14206	1238	96.67%	1147.50%
3	16073	15475	13425	96.28%	115.27%
6	15251	14791	1215	96.98%	1217.37%
7	15798	15382	13979	97.37%	110.04%
8	16139	15570	15205	96.47%	102.40%
9	15319	14891	12935	97.21%	115.12%
10	14958	14437	12548	96.52%	115.05%
	15357	14860.625	10343.5	96.77%	379.82%

Table 106 RF-V203-KeyFrame-Deletion Statistics

#Iterations	Total No. of kfs to pass SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	98	53	414	318	54.08%	16.67%	12.80%	23.67%
2	116	70	433	28	60.34%	250%	16.17%	26.79%
3	103	63	467	366	61.17%	17.21%	13.49%	22.06%
4	107	60	469	363	56.07%	16.53%	12.79%	22.81%
5	107	63	454	28	58.88%	225%	13.88%	23.57%
6	114	71	450	30	62.28%	236.67%	15.78%	25.33%
7	106	56	473	368	52.83%	15.22%	11.84%	22.41%
8	103	63	489	388	61.17%	16.24%	12.88%	21.06%
9	105	58	445	342	55.24%	16.96%	13.03%	23.60%
10	99	54	436	339	54.55%	15.93%	12.39%	22.71%
	105.8	61.1	453	257	57.66%	82.64%	13.50%	23.40%

Table 107 RF-V203-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	4.74	12.89	18.72	96.74	11.5	142.9
2	5.112	12.99	20.2	120.6	14.13	171.3
3	4.963	12.36	20.61	122.1	15.8	173.8
4	5.015	14.06	21.65	126.3	16.06	180.3
5	5.125	14.02	23.61	128.4	16.57	185.3
6	5.244	13.4	24.31	132.2	18.15	191.8
7	4.771	12.81	21.93	107.7	14.4	159.7
8	5.261	14.39	24.84	138	19.72	200.2
9	5.048	12.61	23.82	138.7	18.27	197
10	4.971	13.56	23.21	121.6	15.57	177.1
	5.025	13.309	22.29	123.234	16.017	177.94

Table 108 RF-V203-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	12.92	2.683	4.13	0.074	22.3
2	13.07	2.796	4.489	0.084	23.05
3	10.54	2.496	4.774	0.077	20.1
4	11.6	2.729	5.067	0.069	22.01
5	11.7	2.857	4.675	0.081	21.9
6	11.31	2.67	4.852	0.083	21.1
7	11.48	2.628	4.455	0.078	21.26
8	11.18	2.75	4.771	0.078	21.14
9	10.99	2.506	4.774	0.069	20.8
10	11.08	2.664	4.58	0.072	20.77
	11.587	2.6779	4.6567	0.0765	21.443

Discussion

The deletion scheme with thread safe reference counting with mutexes enabled safe deletion of keyframes as well as map points for all the 11 sequences present in the EuRoC dataset. However, reference counting affects performance to a small degree, the number of keyframes that are processed becomes lesser. This trend extends to the keyframes marked bad as well as the keyframes within the map. All measurements for execution times are in milliseconds.

Map point statistics conform to the same trend with a decrease in the number of map points marked bad and number of map points within the map. A key difference between keyframe and map point deletion is that most of the map points marked bad are deleted while the same cannot be said for keyframes. The percentage of deletion for map points stands at an average of 97.31%(with respect to map points marked bad) for all the datasets while it is at 69.57% for keyframe deletion (with respect to keyframes marked bad).

The time taken by the tracking and local mapping threads also show an increase in time to perform various computations. In short, the reference counting does incur a small performance penalty, but this means that the dynamically allocated memory is freed. To give a rough estimate of the memory involved, the deallocation adds up to roughly 10 Mb of data averaged across all the datasets, where the average length of a dataset would be about three minutes.

ORBSLAM3 Deletion Statistics with Thread Safe Reference Counting using CAS.

ORBSLAM3 was profiled on the EuRoC dataset with an Intel i7-11857G with 32 gigabytes of memory. The EuRoC dataset consists of 11 sequences namely – Machine Hall 01, Machine Hall 02, Machine Hall 03, Machine Hall 04, Machine Hall 05, Vicon Room 1 01, Vicon Room 1 02, Vicon Room 1 03, Vicon Room 2 01, Vicon Room 2 02, Vicon Room 2 03. The times for measurements pertaining to function execution times are in milliseconds. The results are discussed at the end of this section.

The calculations for all quantities are averaged across 10 iterations of each sequence. The following tables represent the number of keyframes, and map points processed and marked for deletion. It also shows the number of deleted keyframes and map points along with related statistics. Execution times of the local mapping, loop closing, and the tracking thread are also tabulated below. These statistics were captured to evaluate both the reference counting implementations.

The following consists of some abbreviated terms that are used in the tables below.

(SBF expands to SetBadFlag)

(kfs/KFs expands to KeyFrames) and (MP expands to MapPoint)

(LM expands to Local Mapping)

(LBA expands to Local Bundle Adjustment)

(No. expands to Number)

CAS-MH01

Table 109 CAS-MH01-General Statistics

#Iterations	Total No. of kfs to pass SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	96	63	382	287	65.63%	21.95%	16.49%	25.13%
2	95	58	375	281	61.05%	20.64%	15.47%	25.33%
3	91	57	378	288	62.64%	19.79%	15.08%	24.07%
4	98	66	379	282	67.35%	23.40%	17.41%	25.86%
5	114	78	413	300	68.42%	26%	18.89%	27.60%
6	121	87	423	303	71.90%	28.71%	20.57%	28.61%
7	111	74	413	303	66.67%	24.42%	17.92%	26.88%
8	116	82	422	307	70.69%	26.71%	19.43%	27.49%
9	128	89	429	302	69.53%	29.47%	20.75%	29.84%
10	114	82	420	307	71.93%	26.71%	19.52%	27.14%
	108.4	73.6	403.4	296	67.58%	24.78%	18.15%	26.79%

Table 110 CAS-MH01-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion	Percentage of deletion (wrt mappoints in maps)
1	15173	14937	11112	98.44%	134.42%
2	15098	14818	10732	98.15%	138.07%
3	15358	15097	10995	98.30%	137.31%
4	15338	15098	10934	98.44%	138.08%
5	15581	15341	11616	98.46%	132.07%
6	16384	16172	11801	98.71%	137.04%
7	15876	15620	11707	98.39%	133.42%
8	16083	15870	11688	98.68%	135.78%
9	15867	15644	11715	98.59%	133.54%
10	15956	15698	11798	98.38%	133.06%
	15671.4	15429.5	11409.8	98.45%	135.28%

Table 111 CAS-MH01-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	7.84634	9.99376	43.39936	292.08118	31.9392	383.65213
2	7.7015	9.8845	42.634	287.25	31.207	377.07
3	7.7607	9.9408	41.983	292.43	31.425	381.71
4	7.5846	9.8727	42.109	282.28	31.274	371.55
5	7.0852	8.6023	36.699	260.61	27.889	339.56
6	7.0999	8.2388	35.805	264.92	26.873	341.6
7	7.0264	8.4472	35.639	259.84	28.116	337.74
8	6.9776	8.3933	35.704	258.73	27.684	336.18
9	6.9506	8.1609	35.215	252.05	27.542	328.67
10	6.9941	8.3583	35.533	259.06	26.982	335.64
	7.302694	8.989256	38.472036	270.925118	29.09312	353.337213

Table 112 CAS-MH01-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	13.10401	2.79817	8.00286	0.08761	26.1692
2	12.59054	2.72079	7.64845	0.08099	25.07901
3	12.70152	2.67262	7.66846	0.07917	25.16347
4	12.43075	2.67106	7.61354	0.07319	24.79148
5	11.63835	2.26679	6.54558	0.06492	22.35536
6	11.81501	2.20729	6.54895	0.0599	22.47866
7	11.83249	2.19488	6.28156	0.05735	22.17294
8	11.76367	2.21286	6.45698	0.06052	22.31538
9	11.73204	2.20569	6.39935	0.06169	22.18818
10	11.82608	2.20182	6.39762	0.06438	22.29146
	12.143446	2.415197	6.956335	0.068972	23.500514

CAS-MH02

Table 113 CAS-MH02-General Statistics

#Iterations	Total No. of kfs to pass SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	84	57	349	266	67.86%	21.43%	16.33%	24.07%
2	95	67	361	267	70.53%	25.09%	18.56%	26.32%
3	98	73	361	264	74.49%	27.65%	20.22%	27.15%
4	91	58	360	270	63.74%	21.48%	16.11%	25.28%
5	88	63	351	264	71.59%	23.86%	17.95%	25.07%
6	100	67	366	267	67%	25.09%	18.31%	27.32%
7	92	62	362	271	67.39%	22.88%	17.13%	25.41%
8	91	58	350	260	63.74%	22.31%	16.57%	26%
9	86	59	355	270	68.60%	21.85%	16.62%	24.23%
10	92	59	352	261	64.13%	22.61%	16.76%	26.14%
	91.7	62.3	356.7	266	67.91%	23.43%	17.46%	25.70%

Table 114 CAS-MH02-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion	Percentage of deletion (wrt mappoints in maps)
1	14268	14020	10183	98.26%	137.68%
2	14082	13846	10208	98.32%	135.64%
3	14470	14238	10362	98.40%	137.41%
4	14374	14184	10292	98.68%	137.82%
5	14694	14482	10240	98.56%	141.43%
6	14633	14357	10363	98.11%	138.54%
7	14444	14219	10384	98.44%	136.93%
8	14146	13902	10170	98.28%	136.70%
9	14722	14490	10356	98.42%	139.92%
10	14216	13999	10141	98.47%	138.04%
	14404.9	14173.7	10269.9	98.39%	138.01%

Table 115 CAS-MH02-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	6.477	8.2468	34.841	271.22	23.838	342.48
2	6.1288	8.0502	34.076	256.07	22.528	325.38
3	6.2916	8.1664	34.997	265.92	24.218	338.01
4	6.3417	8.0519	34.48	264.66	23.319	335.28
5	6.2816	8.2253	34.842	266.04	24.07	337.83
6	6.1227	8.277	33.732	252.9	23.018	322.61
7	6.207	8.2077	34.353	261.74	22.763	331.73
8	6.3845	8.2167	35.008	272.5	24.904	345.39
9	6.3258	8.3502	34.856	272.63	23.689	344.25
10	6.3787	8.2013	34.843	270.2	24.807	342.83
	6.29394	8.19935	34.6028	265.388	23.7154	336.579

Table 116 CAS-MH02-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	11.62133	2.13892	5.26189	0.05306	20.70098
2	11.68561	2.09438	5.07925	0.05467	20.55766
3	11.51226	2.12915	5.40957	0.05225	20.73657
4	11.60353	2.11807	5.15539	0.05306	20.549
5	11.57167	2.1468	5.39305	0.05369	20.80101
6	11.57768	2.12666	5.08935	0.05321	20.45807
7	11.66932	2.15718	5.1563	0.05681	20.66406
8	11.56302	2.16376	5.32574	0.05529	20.74222
9	11.57045	2.13908	5.38872	0.05209	20.7867
10	11.55644	2.1267	5.25313	0.05178	20.63373
	11.593131	2.13407	5.251239	0.053591	20.663

Table 117 CAS-MH03-General Statistics

#Iterations	Total No. of kfs to pass SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	88	30	339	252	34.09%	11.90%	8.85%	25.96%
2	88	32	331	244	36.36%	13.11%	9.67%	26.59%
3	88	31	336	249	35.23%	12.45%	9.23%	26.19%
4	90	36	339	250	40%	14.40%	10.62%	26.55%
5	88	31	341	254	35.23%	12.20%	9.09%	25.81%
6	78	24	326	249	30.77%	9.64%	7.36%	23.93%
7	84	32	332	249	38.10%	12.85%	9.64%	25.30%
8	89	33	335	247	37.08%	13.36%	9.85%	26.57%
9	79	32	327	249	40.51%	12.85%	9.79%	24.16%
10	88	33	338	251	37.50%	13.15%	9.76%	26.04%
	86	31.4	334.4	249.4	36.49%	12.59%	9.39%	25.71%

Table 118 CAS-MH03-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion	Percentage of deletion (wrt mappoints in maps)
1	13123	12744	9158	97.11%	139.16%
2	12764	12363	8922	96.86%	138.57%
3	13046	12616	9063	96.70%	139.20%
4	13156	12769	9032	97.06%	141.38%
5	13162	12786	9202	97.14%	138.95%
6	12974	12572	9051	96.90%	138.90%
7	12838	12440	9018	96.90%	137.95%
8	12848	12427	8954	96.72%	138.79%
9	13004	12610	8966	96.97%	140.64%
10	13032	12652	9148	97.08%	138.30%
	12994.7	12597.9	9051.4	96.95%	139.18%

Table 119 CAS-MH03-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	6.1936	8.5633	29.904	234.54	24.587	302.34
2	6.22943	8.70367	29.70363	228.50994	24.60521	296.30183
3	6.11288	8.81031	29.89542	229.28814	24.67669	297.3501
4	6.2257	8.76631	29.96739	231.78238	24.72801	300.03456
5	6.2872	8.66367	30.10575	235.79435	24.79551	304.19572
6	6.08938	9.11154	29.72824	227.83976	24.05367	295.35613
7	6.13785	8.78522	29.74588	224.25382	23.83688	291.34185
8	6.12269	8.63122	29.6417	231.60228	24.60592	299.15293
9	6.08101	8.65488	29.55051	221.59282	23.59001	288.04687
10	6.15272	8.65945	29.87102	229.63398	24.21369	297.10532
	6.163246	8.734957	29.811354	229.483747	24.369259	297.122531

Table 120 CAS-MH03-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	11.5468	2.1042	5.01741	0.05369	20.65188
2	11.62701	2.08094	4.97243	0.04883	20.66505
3	11.63807	2.14785	5.00886	0.05176	20.7837
4	11.55386	2.13883	5.08563	0.05678	20.76796
5	11.51689	2.13693	5.08995	0.05392	20.72995
6	11.59674	2.11958	5.03632	0.0502	20.71808
7	11.53772	2.13852	5.1115	0.05238	20.76461
8	11.65658	2.12172	5.07693	0.05162	20.86529
9	11.6172	2.12275	5.09854	0.05279	20.82977
10	11.63085	2.11401	5.04304	0.051	20.76336
	11.592172	2.122533	5.054061	0.052297	20.753965

CAS-MH04

Table 121 CAS-MH04-General Statistics

#Iterations	Total number of keyframes to have pass SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	58	45	339	282	77.59%	15.96%	13.27%	17.11%
2	66	50	338	273	75.76%	18.32%	14.79%	19.53%
3	58	46	330	273	79.31%	16.85%	13.94%	17.58%
4	65	53	340	276	81.54%	19.20%	15.59%	19.12%
5	60	56	408	349	93.33%	16.05%	13.73%	14.71%
6	60	48	337	278	80%	17.27%	14.24%	17.80%
7	63	52	336	274	82.54%	18.98%	15.48%	18.75%
8	60	51	372	313	85%	16.29%	13.71%	16.13%
9	71	56	342	272	78.87%	20.59%	16.37%	20.76%
	62.33	50.77	349.11	287.77	81.55%	17.72%	14.57%	17.94%

Table 122 CAS-MH04-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion	Percentage of deletion (wrt mappoints in maps)
1	11261	11043	11534	98.06%	95.74%
2	11064	10828	11072	97.87%	97.80%
3	10973	10729	11161	97.78%	96.13%
4	11205	10976	11206	97.96%	97.95%
5	12849	12634	16124	98.33%	78.36%
6	11161	10938	11294	98.00%	96.85%
7	10902	10671	11058	97.88%	96.50%
8	11837	11617	13748	98.14%	84.50%
9	11331	11082	11045	97.80%	100.33%
	11398.11111	11168.66667	12026.88889	97.98%	93.79%

Table 123 CAS-MH04-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	5.00482	9.06371	23.69072	155.93978	19.12783	211.85406
2	4.97168	8.94824	23.58515	157.00637	18.8275	212.3579
3	4.97126	9.11251	23.65581	153.74611	18.40243	208.90429
4	5.00146	8.86599	23.32477	154.02396	18.69961	208.99273
5	4.63751	9.50098	22.08595	120.9925	19.53869	176.11712
6	5.01847	8.97727	23.57664	153.61548	19.39912	209.62146
7	4.99029	8.90225	23.32113	153.71111	18.82551	208.78303
8	4.70332	8.98244	22.95054	131.75178	19.53983	187.16987
9	5.00354	8.91337	23.37334	151.94182	19.20992	207.50088
	4.922483333	9.02964	23.28489444	148.08099	19.06338222	203.4779267

Table 124 CAS-MH04-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	11.35759	2.22732	4.62254	0.04922	20.08602
2	11.54264	2.22482	4.62324	0.05652	20.31558
3	11.63682	2.15154	4.63774	0.05233	20.32295
4	11.39663	2.17956	4.60339	0.05182	20.06933
5	11.5157	2.16576	4.57649	0.05629	20.22466
6	11.29228	2.2136	4.5801	0.0482	19.93193
7	11.29228	2.2136	4.5801	0.0482	19.93193
8	11.64775	2.19423	4.62625	0.05351	20.36892
9	11.60889	2.10475	4.46513	0.05231	20.11635
10	11.35938	2.1507	4.53179	0.05184	19.92161
	11.464996	2.182588	4.584677	0.052024	20.128928

CAS-MH05

Table 125 CAS-MH05-General Statistics

#Iterations	Total No. of kfs to pass SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	103	75	411	310	72.82%	24.19%	18.25%	25.06%
2	98	78	410	313	79.59%	24.92%	19.02%	23.90%
3	100	82	409	311	82%	26.37%	20.05%	24.45%
4	93	75	387	295	80.65%	25.42%	19.38%	24.03%
5	100	78	412	313	78%	24.92%	18.93%	24.27%
6	95	76	388	294	80%	25.85%	19.59%	24.48%
7	108	84	415	308	77.78%	27.27%	20.24%	26.02%
8	93	76	391	299	81.72%	25.42%	19.44%	23.79%
9	111	91	428	318	81.98%	28.62%	21.26%	25.93%
10	102	79	428	329	77.45%	24.01%	18.46%	23.83%
	100.3	79.4	407.9	309	79.20%	25.70%	19.46%	24.58%

Table 126 CAS-MH05-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion	Percentage of deletion (wrt mappoints in maps)
1	12588	12338	12779	98.01%	96.55%
2	12777	12572	12635	98.40%	99.50%
3	12366	12189	12772	98.57%	95.44%
4	12386	12192	11784	98.43%	103.46%
5	12823	12598	12647	98.25%	99.61%
6	12521	12264	11841	97.95%	103.57%
7	12506	12280	12491	98.19%	98.31%
8	12435	12225	11829	98.31%	103.35%
9	12874	12667	13091	98.39%	96.76%
10	12936	12707	13295	98.23%	95.58%
	12621.2	12403.2	12516.4	98.27%	99.21%

Table 127 CAS-MH05-Local Mapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	5.479	8.6269	24.999	154.73	19.036	212.03
2	5.4031	8.7781	24.91	157.19	19.398	214.87
3	5.3767	8.6715	24.755	149.71	19.467	207.17
4	5.4938	8.4858	25.666	160.71	22.211	221.3
5	5.4216	8.6821	25.159	156.17	19.552	214.12
6	5.4658	8.6039	25.706	158.14	21.956	218.94
7	5.3198	8.444	24.618	154.99	19.154	211.69
8	5.5277	8.6234	25.823	161.62	22.477	223.11
9	5.2896	8.7622	24.622	142.16	18.242	197.57
10	5.267	8.6647	24.43	145.19	18.343	201.13
	5.40441	8.63426	25.0688	154.061	19.9836	212.193

Table 128 CAS-MH05-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	11.04767	2.27002	4.77046	0.05317	20.33013
2	11.16152	2.27989	4.71227	0.05801	20.37906
3	11.24615	2.33316	4.82132	0.05834	20.67857
4	11.24615	2.33316	4.82132	0.05834	20.67857
5	11.01413	2.30037	4.79545	0.06526	20.34746
6	11.01035	2.22718	4.58749	0.05664	20.01574
7	11.1709	2.25471	4.78457	0.05744	20.48011
8	11.09203	2.27146	4.69951	0.0544	20.2475
9	11.28363	2.25726	4.57887	0.05719	20.38217
10	10.95767	2.23154	4.44234	0.06198	19.82482
	11.12302	2.275875	4.70136	0.058077	20.336413

CAS-V101

Table 129 CAS-V101-General Statistics

#Iterations	Total No. of kfs to pass SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	117	65	346	230	55.56%	28.26%	18.79%	33.82%
2	112	66	345	234	58.93%	28.21%	19.13%	32.46%
3	114	67	348	235	58.77%	28.51%	19.25%	32.76%
4	121	75	355	235	61.98%	31.91%	21.13%	34.08%
5	112	63	342	231	56.25%	27.27%	18.42%	32.75%
6	118	69	348	231	58.47%	29.87%	19.83%	33.91%
7	123	73	350	228	59.35%	32.02%	20.86%	35.14%
8	116	67	348	233	57.76%	28.76%	19.25%	33.33%
9	119	69	344	226	57.98%	30.53%	20.06%	34.59%
10	116	67	346	231	57.76%	29.00%	19.36%	33.53%
	116.8	68.1	347.2	231.4	58.28%	29.43%	19.61%	33.64%

Table 130 CAS-V101-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion	Percentage of deletion (wrt mappoints in maps)
1	13370	12980	9556	97.08%	135.83%
2	13323	12932	9552	97.07%	135.39%
3	13240	12863	9529	97.15%	134.99%
4	13499	13104	9683	97.07%	135.33%
5	13264	12861	9492	96.96%	135.49%
6	13284	12879	9546	96.95%	134.92%
7	13231	12843	9493	97.07%	135.29%
8	13486	13107	9613	97.19%	136.35%
9	13396	13020	9298	97.19%	140.03%
10	13164	12771	9517	97.01%	134.19%
	13325.7	12936	9527.9	97.08%	135.78%

Table 131 CAS-V101-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	7.188	10.88	31.54	278.5	34.08	360.5
2	7.154	10.76	31.17	273.4	34.3	355.1
3	7.287	11.05	31.51	279.2	33.84	361.2
4	7.227	11.07	31.37	275.7	34.26	358
5	7.171	10.88	31.18	276.9	33.81	358.2
6	7.201	10.75	31.22	273.5	34.71	355.8
7	7.187	10.82	31.56	276.6	34.07	358.6
8	7.298	10.9	31.39	274.6	34.67	357.3
9	7.217	11.11	31.47	273.1	35.5	356.8
10	7.312	10.91	31.61	276	35	359.2
	7.2242	10.913	31.402	275.75	34.424	358.07

Table 132 CAS-V101-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	10.63	2.411	6.159	0.059	21.44
2	10.61	2.417	6.154	0.064	21.41
3	10.59	2.421	6.157	0.07	21.39
4	10.56	2.424	6.138	0.065	21.34
5	10.62	2.393	6.055	0.057	21.27
6	10.56	2.373	6.092	0.06	21.24
7	10.55	2.399	6.09	0.062	21.27
8	10.54	2.39	6.072	0.06	21.22
9	10.57	2.375	5.989	0.06	21.13
10	10.68	2.411	6.01	0.062	21.31
	10.591	2.4014	6.0916	0.0619	21.302

CAS-V102

Table 133 CAS-V102-General Statistics

#Iterations	Total No. of kfs to pass SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	48	24	301	254	50%	9.45%	7.97%	15.95%
2	49	12	272	225	24.49%	5.33%	4.41%	18.01%
3	49	22	302	255	44.90%	8.63%	7.28%	16.23%
4	44	12	267	225	27.27%	5.33%	4.49%	16.48%
5	55	18	296	237	32.73%	7.59%	6.08%	18.58%
6	34	9	267	234	26.47%	3.85%	3.37%	12.73%
7	52	23	303	252	44.23%	9.13%	7.59%	17.16%
	47.28	17.14	286.85	240.28	0.357	0.070	0.058	0.164

Table 134 CAS-V102-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion	Percentage of deletion (wrt mappoints in maps)
1	11153	10695	10486	95.89%	101.99%
2	11015	10634	10427	96.54%	101.99%
3	10631	10354	8665	97.39%	119.49%
4	10610	10346	8951	97.51%	115.58%
5	10735	10348	10603	96.39%	97.60%
6	10069	9783	8716	97.16%	112.24%
7	10720	10426	9195	97.26%	113.39%
8	10607	10338	9027	97.46%	114.52%
9	11029	10602	10647	96.13%	99.58%
10	10944	10518	10479	96.11%	100.37%
	10751.3	10404.4	9719.6	96.79%	107.68%

Table 135 CAS-V102-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	4.937	11.57	20.49	155.6	13.04	201.8
2	5.042	11.05	20.83	161.2	14.11	209.4
3	5.303	10.12	23.62	197	17.21	251
4	5.31	10.42	23.65	197.9	16.9	252.3
5	4.991	11.18	20.29	158.2	13.43	205.9
6	5.276	10.29	23.34	188	16.21	241.7
7	5.451	10.92	24.08	155.1	19.67	213
8	5.314	10.5	23.25	189.5	16.26	243.3
9	4.984	11.44	20.26	154.3	12.82	201.7
10	5.026	10.89	20.51	155.6	13.81	203.7
	5.1634	10.838	22.032	171.24	15.346	222.38

Table 136 CAS-V102-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	11.04	2.425	4.172	0.066	20.14
2	10.96	2.381	4.289	0.053	20.11
3	10.51	2.182	4.364	0.049	19.46
4	10.46	2.167	4.397	0.05	19.45
5	11.05	2.398	4.278	0.052	20.25
6	10.72	2.23	4.379	0.056	19.75
7	11.13	2.284	4.482	0.051	20.48
8	10.55	2.162	4.427	0.051	19.58
9	11.04	2.416	4.237	0.055	20.21
10	11	2.367	4.27	0.052	20.15
	10.846	2.3012	4.3295	0.0535	19.958

CAS-V103

Table 137 CAS-V103-General Statistics

#Iterations	Total No. of kfs to pass SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	115	61	420	309	53.04%	19.74%	14.52%	27.38%
2	115	70	484	373	60.87%	18.77%	14.46%	23.76%
3	90	68	472	384	75.56%	17.71%	14.41%	19.07%
4	98	51	395	298	52.04%	17.11%	12.91%	24.81%
5	105	57	424	291	54.29%	19.59%	13.44%	24.76%
6	125	86	521	368	68.80%	23.37%	16.51%	23.99%
7	135	98	516	382	72.59%	25.65%	18.99%	26.16%
8	94	61	424	332	64.89%	18.37%	14.39%	22.17%
9	106	79	499	394	74.53%	20.05%	15.83%	21.24%
10	97	64	462	366	65.98%	17.49%	13.85%	21.00%
	108	69.5	461.7	349.7	64.26%	19.79%	14.93%	23.43%

Table 138 CAS-V103-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion	Percentage of deletion (wrt mappoints in maps)
1	15289	14686	11089	96.06%	132.44%
2	16326	15657	12912	95.90%	121.26%
3	17115	16221	13464	94.78%	120.48%
4	16042	15586	10808	97.16%	144.21%
5	16216	15450	10152	95.28%	152.19%
6	17589	16851	12563	95.80%	134.13%
7	16459	15563	13749	94.56%	113.19%
8	16604	15753	11849	94.87%	132.95%
9	16539	15853	14667	95.85%	108.09%
10	16722	15847	13205	94.77%	120.01%
	16490.1	15746.7	12445.8	95.50%	127.89%

Table 139 CAS-V103-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	4.657	11.4	21.35	131.6	15.82	183.2
2	4.28	10.95	18.69	113.9	13.24	159.6
3	4.277	13.07	19.22	106.1	12.87	153.9
4	5.054	11.14	22.52	140.6	17.01	195.6
5	4.677	12.7	20.95	124.2	14.48	174.5
6	4.281	11.24	19.49	97.05	14.63	145.7
7	4.204	11.47	16.61	88.07	9.82	128.7
8	4.688	12.73	20.5	118.7	14.28	168.8
9	4.3	11.73	18.82	98.83	13.9	146.7
10	4.404	12.96	19.54	107.4	13.42	156
	4.4822	11.939	19.769	112.645	13.947	161.27

Table 140 CAS-V103-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	10.71	2.153	4.091	0.058	19.55
2	10.72	2.176	3.677	0.062	19.19
3	11.14	2.285	3.651	0.062	19.75
4	10.78	2.32	4.068	0.054	19.72
5	11.15	2.366	3.655	0.051	19.82
6	10.47	2.078	3.682	0.06	18.88
7	10.6	2.272	3.06	0.056	18.5
8	11.59	2.546	3.423	0.056	20.2
9	10.61	2.38	4.005	0.054	19.67
10	11.27	2.387	3.811	0.065	20.16
	10.904	2.2963	3.7123	0.0578	19.544

CAS-V201

Table 141 CAS-V201-General Statistics

#Iterations	Total No. of kfs to pass SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	168	158	435	269	94.05%	58.74%	36.32%	38.62%
2	155	147	411	257	94.84%	57.20%	35.77%	37.71%
3	158	149	410	253	94.30%	58.89%	36.34%	38.54%
4	158	147	409	252	93.04%	58.33%	35.94%	38.63%
5	155	146	407	253	94.19%	57.71%	35.87%	38.08%
6	175	162	451	279	92.57%	58.06%	35.92%	38.80%
7	163	153	408	246	93.87%	62.20%	37.50%	39.95%
8	152	143	405	254	94.08%	56.30%	35.31%	37.53%
9	155	148	406	252	95.48%	58.73%	36.45%	38.18%
	159.8	150.33	415.77	257.22	94.05%	58.46%	36.16%	38.45%

Table 142 CAS-V201-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion	Percentage of deletion (wrt mappoints in maps)
1	11470	11138	11039	97.11%	100.90%
2	11332	11002	11341	97.09%	97.01%
3	11192	10898	11236	97.37%	96.99%
4	11079	10765	11064	97.17%	97.30%
5	11069	10755	11041	97.16%	97.41%
6	11560	11238	11090	97.21%	101.33%
7	606	491	1038	81.02%	47.30%
8	11315	10992	10943	97.15%	100.45%
9	11154	10833	11313	97.12%	95.76%
10	11325	11022	10993	97.32%	100.26%
	10210.2	9913.4	10109.8	95.57%	93.47%

Table 143 CAS-V201-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	6.054	10.55	24.25	142.6	20.91	202.9
2	6.15	10.91	25.13	155	22.81	219.2
3	6.243	10.85	25.48	159.3	23.13	224.2
4	6.242	10.87	24.99	156.5	23.52	221.3
5	6.245	10.72	25.03	156.2	23.43	220.8
6	5.807	10.53	23.24	140	19.54	198.5
7	6.231	10.69	25.07	157.4	22.72	221.3
8	6.193	10.73	25.35	159.8	23.07	224.3
9	6.302	11.01	25.16	159	22.73	223.4
	6.163	10.76	24.85	153.97	22.42	217.32

Table 144 CAS-V201-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	10.94	2.564	4.457	0.064	20.1
2	11.06	2.652	4.571	0.06	20.39
3	10.98	2.618	4.533	0.057	20.25
4	10.93	2.619	4.609	0.055	20.27
5	10.96	2.632	4.678	0.058	20.37
6	10.98	2.54	4.087	0.053	19.72
7	10.98	2.624	4.63	0.061	20.36
8	11	2.673	4.642	0.061	20.43
9	10.97	2.663	4.611	0.055	20.34
	10.97777778	2.620555556	4.535333333	0.05822222222	20.24777778

CAS-V202

Table 145 CAS-V202-General Statistics

#Iterations	Total No. of kfs to pass SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	89	52	375	287	58.43%	18.12%	13.87%	23.73%
2	93	55	389	298	59.14%	18.46%	14.14%	23.91%
3	82	63	483	404	76.83%	15.59%	13.04%	16.98%
4	89	67	520	434	75.28%	15.44%	12.88%	17.12%
5	49	48	232	50	97.96%	96%	20.69%	21.12%
6	111	70	411	302	63.06%	23.18%	17.03%	27.01%
7	83	52	362	280	62.65%	18.57%	14.36%	22.93%
8	86	52	371	286	60.47%	18.18%	14.02%	23.18%
9	90	53	363	275	58.89%	19.27%	14.60%	24.79%
10	91	51	382	294	56.04%	17.35%	13.35%	23.82%
	86.3	56.3	388.8	291	66.87%	26.02%	14.80%	22.46%

Table 146 CAS-V202-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion	Percentage of deletion (wrt mappoints in maps)
1	13068	12478	11335	95.49%	110.08%
2	12185	11882	11095	97.51%	107.09%
3	15498	15178	15381	97.94%	98.68%
4	16656	16360	15019	98.22%	108.93%
5	7437	6796	3094	91.38%	219.65%
6	12560	12268	10709	97.68%	114.56%
7	12055	11761	10445	97.56%	112.60%
8	12569	12270	10786	97.62%	113.76%
9	12209	11906	10267	97.52%	115.96%
10	12006	11699	11086	97.44%	105.53%
	12624.3	12259.8	10921.7	96.84%	120.68%

Table 147 CAS-V202-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	5.568	11.84	23.96	191.4	21.25	251.3
2	5.549	10.05	23.2	187.4	20.27	244.4
3	5.069	9.988	22.58	139.9	18.9	195.5
4	4.901	10.65	21.94	116.8	15.65	169
5	5.575	9.875	23.61	178.1	20.36	236.2
6	5.86	9.827	25.45	203.8	23.67	266.9
7	5.754	10.12	25.22	196.9	23.47	259.7
8	5.703	9.963	25.19	192.4	23.08	255.3
9	5.501	10.14	23.28	186.2	20.55	244.6
	5.497	10.27	23.82	176.9	20.8	235.8

Table 148 CAS-V202-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	10.46	2.122	5.5	0.056	19.95
2	10.22	2.048	5.749	0.052	19.86
3	10.19	1.953	5.273	0.073	19.31
4	10.37	2.066	4.279	0.069	18.59
5	10.14	2.03	5.648	0.056	19.63
6	10.2	2.057	5.663	0.059	19.7
7	10.23	2.005	5.622	0.057	19.65
8	10.27	1.995	5.399	0.057	19.44
9	10.21	2.057	5.746	0.053	19.82
	10.25444444	2.037	5.431	0.05911111111	19.55

CAS-V203

Table 149 CAS-V203-General Statistics

#Iterations	Total No. of kfs to pass SBF	Total No. of deleted keyframes	Current KF ID (Max)	No. of keyframes within map	Percentage of deletion	Percentage of deletion wrt kfs in map	Percentage of deletion (wrt kfs max mnid)	Percentage of kfs passed SetBadFlag (wrt kfs max mnid)
1	110	69	466	29	62.73%	237.93%	14.81%	23.61%
2	116	67	484	30	57.76%	223.33%	13.84%	23.97%
3	97	56	458	363	57.73%	15.43%	12.23%	21.18%
4	118	69	486	30	58.47%	230%	14.20%	24.28%
5	125	76	466	30	60.80%	253.33%	16.31%	26.82%
6	116	70	450	28	60.34%	250%	15.56%	25.78%
7	105	59	454	352	56.19%	16.76%	13.00%	23.13%
8	126	77	514	30	61.11%	256.67%	14.98%	24.51%
9	115	61	431	317	53.04%	19.24%	14.15%	26.68%
10	106	71	495	25	66.98%	284%	14.34%	21.41%
	113.4	67.5	470.4	123.4	59.52%	178.67%	14.34%	24.14%

Table 150 CAS-V203-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion	Percentage of deletion (wrt mappoints in maps)
1	15719	15196	1200	96.67%	1266.33%
2	15763	15184	1212	96.33%	1252.81%
3	15705	15102	13735	96.16%	109.95%
4	16376	15833	1227	96.68%	1290.38%
5	16310	15758	1236	96.62%	1274.92%
6	15202	14638	1073	96.29%	1364.21%
7	15407	14951	13005	97.04%	114.96%
8	16647	16025	1237	96.26%	1295.47%
9	14754	14233	12309	96.47%	115.63%
10	16782	16092	1178	95.89%	1366.04%
	15866.5	15301.2	4741.2	96.44%	945.07%

Table 151 CAS-V203-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	4.756	12.44	18.06	101.9	14.08	149.8
2	4.684	12.29	17.31	97.21	12.29	142.5
3	4.575	12.35	17.98	99.83	12.97	146
4	4.827	12.16	18.3	107.5	13.02	154.1
5	4.86	12.53	18.77	111.7	13.45	159.5
6	4.859	12.25	18.46	110.2	13.32	157.3
7	4.785	11.97	19.11	110.7	14.51	159.8
8	4.621	11.6	17.61	96.27	12.69	141.4
9	4.93	11.79	19.49	114.4	15.11	164.6
10	4.587	12.05	17.72	98.87	12.45	144.1
	4.7484	12.143	18.281	104.858	13.389	151.91

Table 152 CAS-V203-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	12.96	2.636	4.05	0.059	22.28
2	12.97	2.656	3.954	0.068	22.13
3	13.05	2.609	4.045	0.064	22.29
4	12.1	2.557	4.101	0.067	21.17
5	12.07	2.538	4.031	0.064	21.04
6	12.85	2.576	4.04	0.055	22
7	12.02	2.472	4.405	0.069	21.33
8	12.16	2.513	3.928	0.068	21.05
9	12.77	2.68	4.155	0.067	22.13
10	12.05	2.461	4.077	0.063	21.19
	12.5	2.5698	4.0786	0.0644	21.661

Discussion

The deletion scheme with thread safe reference counting using compare and swap enabled safe deletion of keyframes as well as map points for all the 11 sequences present in the EuRoC dataset.

Generally, this implementation is slightly faster than the mutex locked implementation for most of the quantities measured. Hence, the number of keyframes and map points deleted are slightly more. This observation extends to the computations carried out by the local mapping, tracking as well as loop closing thread. Most of these operations take slightly less time compared to the time taken by their counterparts for the mutex locked reference counting.

A comparison of the metrics from plain ORBSLAM3, and reference counting implemented using mutexes and CAS is given below.

Comparison of Statistics for Mutex and CAS based Reference Counting

The percentage difference of execution times and quantities for different operations are compared with respect to the type of implementation. The times for measurements pertaining to function execution times are in milliseconds.

Table 153 Vanilla-CAS-RF-LocalMapping Statistics

Local Mapping Statistics						
Dataset ID	VANILLA	RF	%difference RF-VAN	CAS	%difference CAS-VAN	% diff CAS RF
MH_01_easy	310.438	346.27	10.91	353.337213	12.92	2.019
MH_02_easy	315.648	342.405	8.132	336.579	6.418	1.716
MH_03_medium	270.194	302.79	11.37	297.122531	9.493	1.889
MH_04_difficult	187.273527	209.112	11.01	203.4779267	8.293	2.731
MH_05_difficult	189.508	213.11	11.72	212.193	11.29	0.431
V1_01_easy	326.48	359.11	9.518	358.07	9.229	0.29
V1_02_medium	213.09	250.17	16	222.38	4.266	11.76
V1_03_difficult	164.09	154.77	5.845	161.27	1.733	4.113
V2_01_easy	185.42	211.14	12.97	195.59	5.338	7.646
V2_02_Medium	215.52	243.66	12.25	235.87	9.016	3.249
V2_03_difficult	138.74	177.94	24.75	151.59	8.851	15.99

Table 154 Vanilla-CAS-RF-Tracking Statistics

	Tracking Statistics					
Dataset ID	VANILLA	%difference RF-VAN	RF	CAS	%difference CAS-VAN	%difference RF CAS
MH_01_easy	21.2	9.29	23.30	23.50	10.14	0.849
MH_02_easy	20.03	7.39	21.57	20.66	3.082	4.315
MH_03_medium	19.84	8.5	21.62	20.7	4.484	4.106
MH_04_difficult	19.88	5.89	21.08	20.12	1.234	4.662
MH_05_difficult	19.79	4.3	20.68	20.33	2.678	1.707
V1_01_easy	20.47	6.88	21.93	21.3	2.678	2.914
V1_02_medium	19.35	6.78	20.71	19.95	3.974	3.738
V1_03_difficult	19.3	1.4	19.59	19.54	3.053	0.255
V2_01_easy	19.21	6.00	20.4	20.24	1.235	0.787
V2_02_Medium	18.95	7.41	20.41	19.55	5.221	4.304
V2_03_difficult	20.64	3.8	21.44	21.61	3.116	0.789

Table 155 Vanilla-CAS-RF-MapPointMap-1 Statistics

	Number of MapPoints within map					
Dataset ID	VANILLA	%difference RF-VAN	RF	% Difference CAS-VAN	%diff RF-CAS	CAS
MH_01_easy	12139.8	6.09	11422.3	6.199	0.109	11409.8
MH_02_easy	10706	3.412	10346.8	4.158	0.745	10269.9
MH_03_medium	9337.6	1.994	9153.2	3.112	1.118	9051.4
MH_04_difficult	12233.9	7.799	11315.5	1.706	6.095	12026.88
MH_05_difficult	12856.6	4.819	12251.6	2.681	2.138	12516.4
V1_01_easy	9790.1	3.519	9451.5	2.714	0.805	9527.9
V1_02_medium	9562.7	1.84	9388.3	1.627	3.467	9719.6
V1_03_difficult	11565.3	2.069	11328.4	7.334	9.4	12445.8
V2_01_easy	11812.3	8.771	10819.7	15.53	6.783	10109.8
V2_02_Medium	11096.2	3.97	10664.2	1.585	2.385	10921.7

Table 156 Vanilla-CAS-RF-MapPoint-2-Deletion Statistics

	Percentage of deletion (MapPoints)		
Dataset ID	RF	CAS	% Difference RF -CAS
MH_01_easy	98.50%	98.45%	0.047
MH_02_easy	98.39%	98.39%	0
MH_03_medium	97.05%	96.95%	0.109
MH_04_difficult	97.84%	97.98%	0.138
MH_05_difficult	98.26%	98.27%	0.015
V1_01_easy	97.16%	97.08%	0.08
V1_02_medium	96.89%	96.79%	0.1
V1_03_difficult	95.68%	95.50%	0.187
V2_01_easy	97.17%	95.57%	1.664
V2_02_Medium	96.70%	97%	0.146
V2_03_difficult	96.78%	96.44%	0.352

Table 157 Vanilla-CAS-RF-MapPointMap-3-Deletion Statistics

Percentage of deletion of MapPoints wrt MapPoints in map			
Dataset ID	CAS	RF	% difference RF CAS
MH_01_easy	135.28%	135.70%	0.31
MH_02_easy	138.01%	137.12%	0.646
MH_03_medium	139.18%	138.05%	0.817
MH_04_difficult	93.79%	96.10%	2.427
MH_05_difficult	99.21%	99.77%	0.559
V1_01_easy	135.78%	134.49%	0.954
V1_02_medium	107.68%	106.78%	0.839
V1_03_difficult	127.89%	216.39%	51.41
V2_01_easy	93.47%	98.92%	5.665
V2_02_Medium	120.68%	112.97%	6.599
V2_03_difficult	945.07%	379.82%	85.32

Table 158 Vanilla-CAS-RF-KeyFrame-1-Deletion Statistics

	Current KF ID (Max)					
Dataset ID	VANILLA	%difference RF-VAN	RF	%difference CAS-VAN	RF-CAS	CAS
MH_01_easy	446.7	9.742	405.2	10.18	0.445	403.4
MH_02_easy	373.1	4.578	356.4	4.494	0.084	356.7
MH_03_medium	354	5.039	336.6	5.694	0.655	334.4
MH_04_difficult	367.1	8.372	337.6	5.023	3.352	349.11
MH_05_difficult	432.1	7.94	399.1	5.761	2.18	407.9
V1_01_easy	374.5	9.159	341.7	7.565	1.596	347.2
V1_02_medium	290.3	2.971	281.8	1.195	1.776	286.85
V1_03_difficult	446.6	3.864	464.2	3.324	0.54	461.7
V2_01_easy	469.6	17.48	394.1	12.15	5.351	415.77
V2_02_Medium	417	11.72	370.8	6.999	4.739	388.8
V2_03_difficult	496.75	9.212	453	5.448	3.768	470.4

Table 159 Vanilla-CAS-RF-KeyFrame-2-Deletion Statistics

	Number of KeyFrames that passed SetBadFlag					
Dataset ID	VANILLA	%Difference RF-VAN	RF	%difference CAS-VAN	CAS-RF	CAS
MH_01_easy	143.7	27.46	109	28	0.551	108.4
MH_02_easy	107.1	16.36	90.9	15.49	0.876	91.7
MH_03_medium	102.6	15.65	87.7	17.6	1.957	86
MH_04_difficult	76.9	23.06	61	20.92	2.162	62.33
MH_05_difficult	118.3	21.63	95.2	16.46	5.217	100.3
V1_01_easy	140	21.16	113.2	18.06	3.13	116.8
V1_02_medium	54.1	22.63	43.1	13.45	9.249	47.28
V1_03_difficult	123.7	20.3	100.9	13.55	6.797	108
V2_01_easy	204.2	36	141.9	24.34	11.91	159.88
V2_02_Medium	115.3	26.63	88.2	28.76	2.177	86.3
V2_03_difficult	140	27.82	105.8	20.99	6.934	113.4

Table 160 Vanilla-CAS-RF-KeyFrame-3-Deletion Statistics

	No. of keyframes within map					
Dataset ID	VANILLA	%Difference RF-VAN	RF	%Difference CAS-VAN	%Difference CAS-RF	CAS
MH_01_easy	304	2.262	297.2	2.666	0.404	296
MH_02_easy	267	0.187	266.5	0.375	0.187	266
MH_03_medium	252.4	0.995	249.9	1.195	0.2	249.4
MH_04_difficult	291.2	4.781	277.6	1.182	3.6	287.7
MH_05_difficult	315.1	3.29	304.9	1.954	1.335	309
V1_01_easy	235.5	2.58	229.5	1.756	0.824	231.4
V1_02_medium	238	0.795	239.9	0.953	0.158	240.28
V1_03_difficult	324.4	1.377	328.9	7.506	6.13	349.7
V2_01_easy	266.9	5.031	253.8	3.693	1.338	257.22
V2_02_Medium	303.9	6.805	283.9	4.336	2.469	291
V2_03_difficult	358.125	2.8103	257	2.5616	1.6645	123.44

Table 161 Vanilla-CAS-RF-KeyFrame-3-Deletion Statistics

	Total No. of deleted keyframes			Percentage of deletion		
Dataset ID	RF	CAS	%Difference	RF	CAS	%Difference
MH_01_easy	73.95	73.6	0.474	67.96%	67.58%	0.56
MH_02_easy	61.9	62.3	0.644	72.20%	67.91%	6.129
MH_03_medium	31.8	31.4	1.265	42.02%	36.49%	14.09
MH_04_difficult	48.4	50.77	4.794	81.92%	81.55%	0.454
MH_05_difficult	77.3	79.4	2.68	83.56%	79.20%	5.359
V1_01_easy	68	68.1	0.146	66.17%	59.50%	10.61
V1_02_medium	70.3	69.5	1.144	59.42%	64.26%	7.826
V1_03_difficult	131	150.33	13.74	93.56%	94.05%	0.522
V2_01_easy	51.7	56.3	8.518	64.64%	66.87%	3.391
V2_02_Medium	61.1	67.5	9.953	64.33%	59.52%	7.767
V2_03_difficult	65.322	70.92	4.3358	69.58%	67.69%	5.6708

Table 162 Vanilla-CAS-RF-KeyFrame-4-Deletion Statistics

	% Deletion wrt KF in map		
Dataset ID	RF	CAS	%Difference
MH_01_easy	25.47	24.78	2.746
MH_02_easy	23.22	23.43	0.9
MH_03_medium	12.73	12.59	1.105
MH_04_difficult	17.44	17.72	1.592
MH_05_difficult	24.49	25.7	4.821
V1_01_easy	29.53	29.43	0.339
V1_02_medium	34.92	19.79	55.3
V1_03_difficult	51.64	58.46	12.38
V2_01_easy	18.23	26.02	35.2
V2_02_Medium	82.64	178.67	73.49
V2_03_difficult	32.031	41.659	18.7873

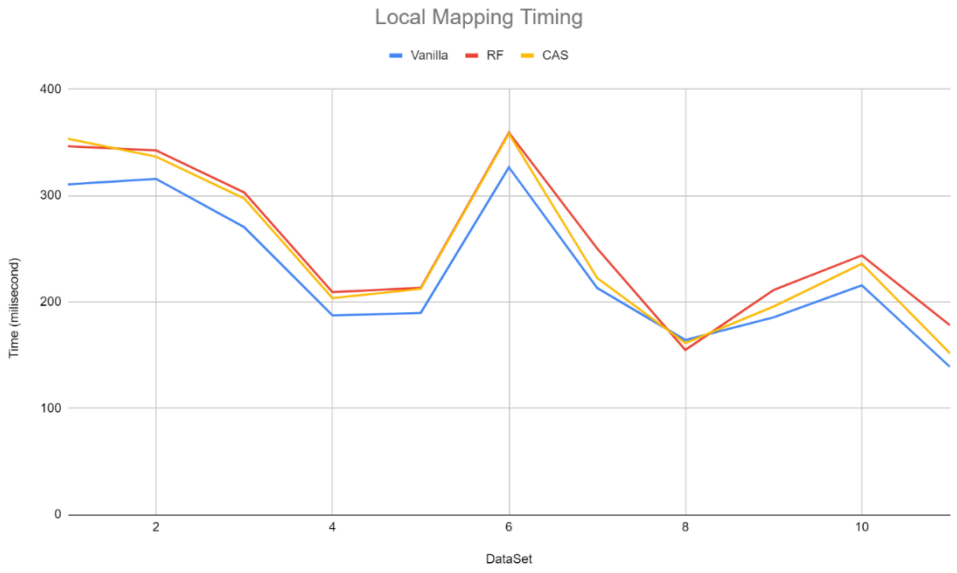
Discussion

All measurements for execution times are in milliseconds. The graph below depicts the local mapping execution times plotted with respect to the 11 datasets. ORBSLAM3 without deletion demarcated by the blue line performs the best taking the least amount of time, followed by the reference counting implemented with CAS (yellow line) and finally reference counting implemented with mutex locks (red line).

This is in line with expectations, as reference counting will slow down the execution of functions due to the constant increments and decrements. Amongst the reference counted implementations, the CAS based reference counting is generally faster.

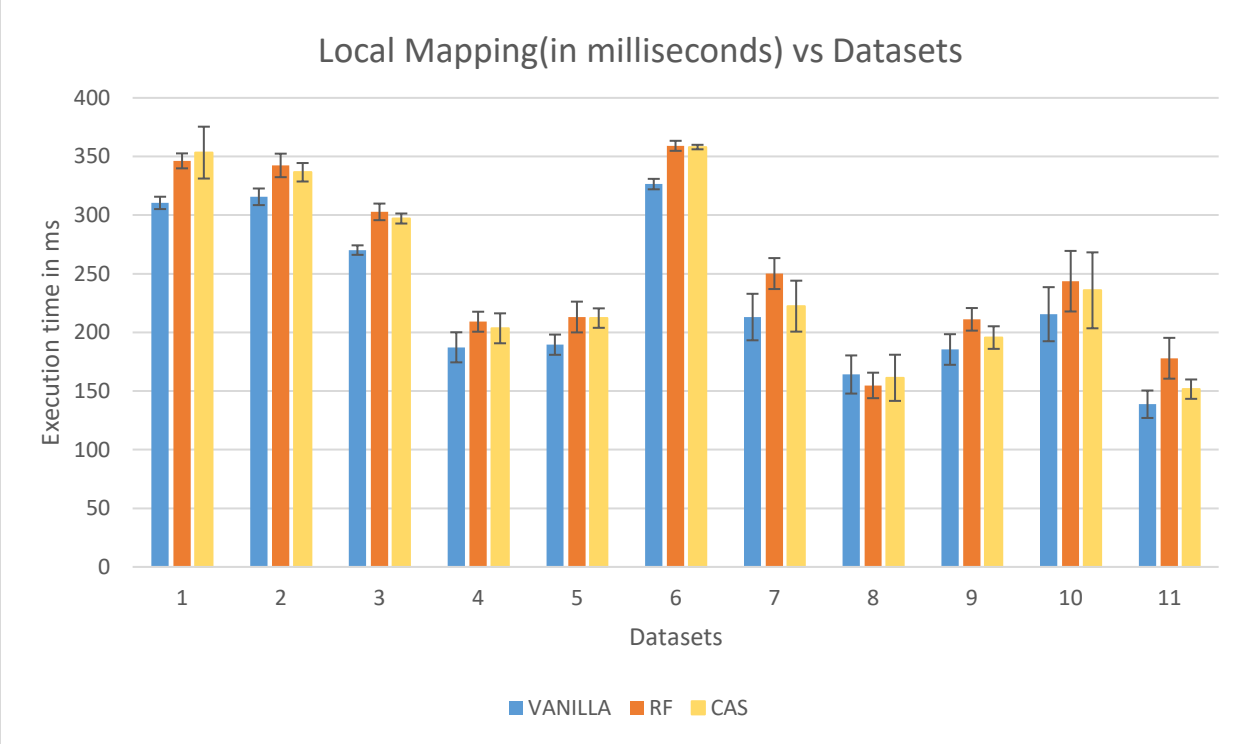
This is not surprising as CAS operations will speed up counting for local mapping operations when compared to using mutexes. The reason behind the dip of the red line at the 8th dataset indicating the best performance amongst all the three implementations must be investigated in the future.

Figure 1 Local Mapping Execution Time wrt Datasets



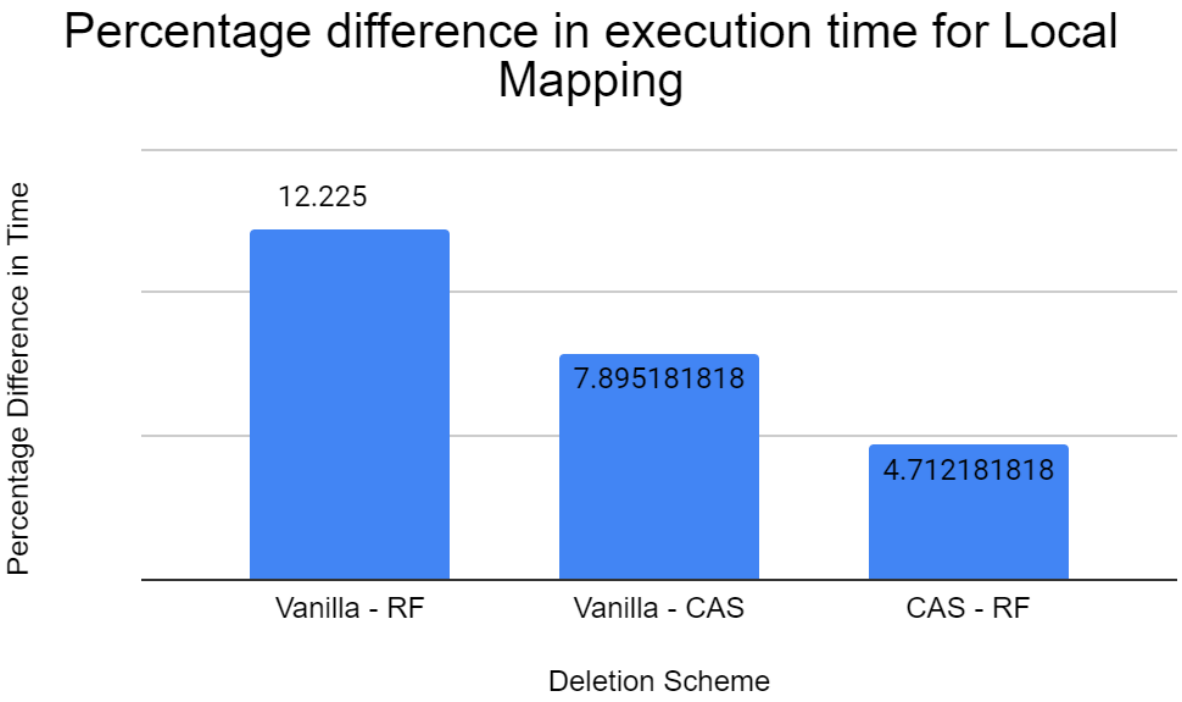
The below figure represents the above figure with error bars as clustered columns.

Figure 2 Local Mapping Execution Time (with error bars) wrt Datasets



The figure below shows the percentage difference of the time taken to execute local mapping for all three different versions of ORBSLAM3. The CAS version is closer to plain ORBSLAM3 in terms of execution time. The mutex locked implementation is the slowest, as it is about 4% slower than the CAS implementation. As mentioned above, this is in line with the expected behavior.

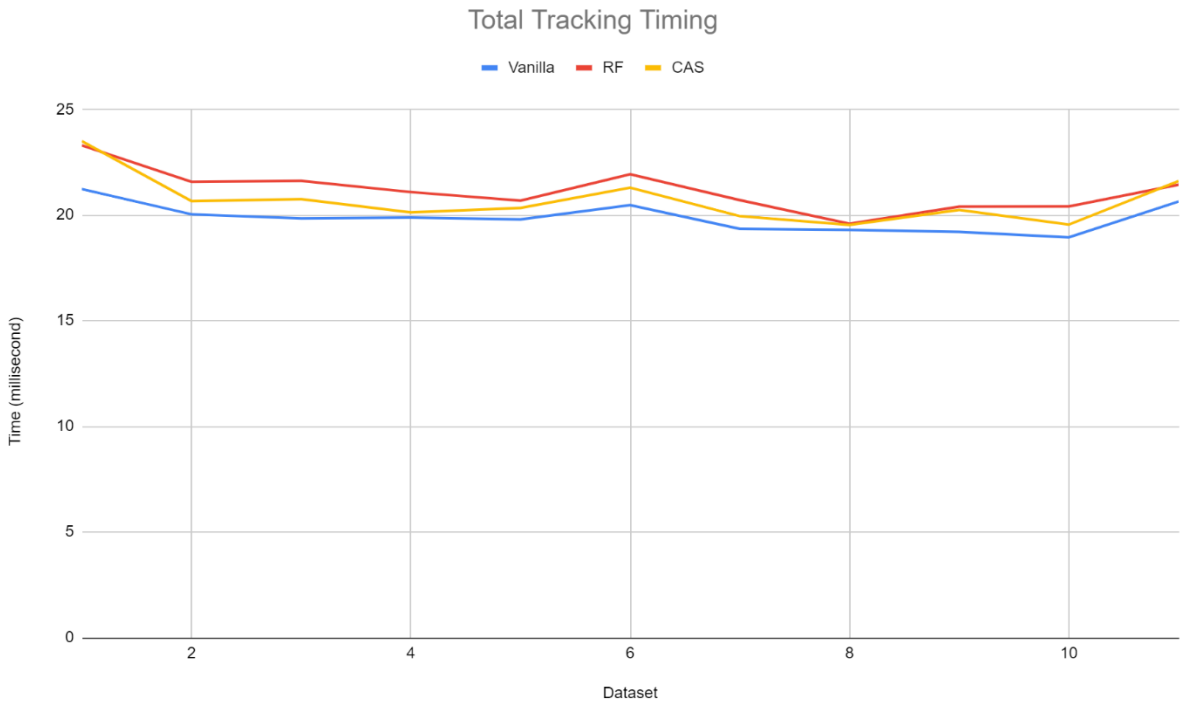
Figure 3 Local Mapping Percentage Difference in Time by Implementation



All measurements for execution times are in milliseconds. The graph below depicts the tracking execution times plotted with respect to the 11 datasets. ORBSLAM3 without deletion demarcated by the blue line performs the best, taking the least amount of time, followed by the reference counting implemented with CAS (yellow line) and finally reference counting implemented with mutex locks (red line).

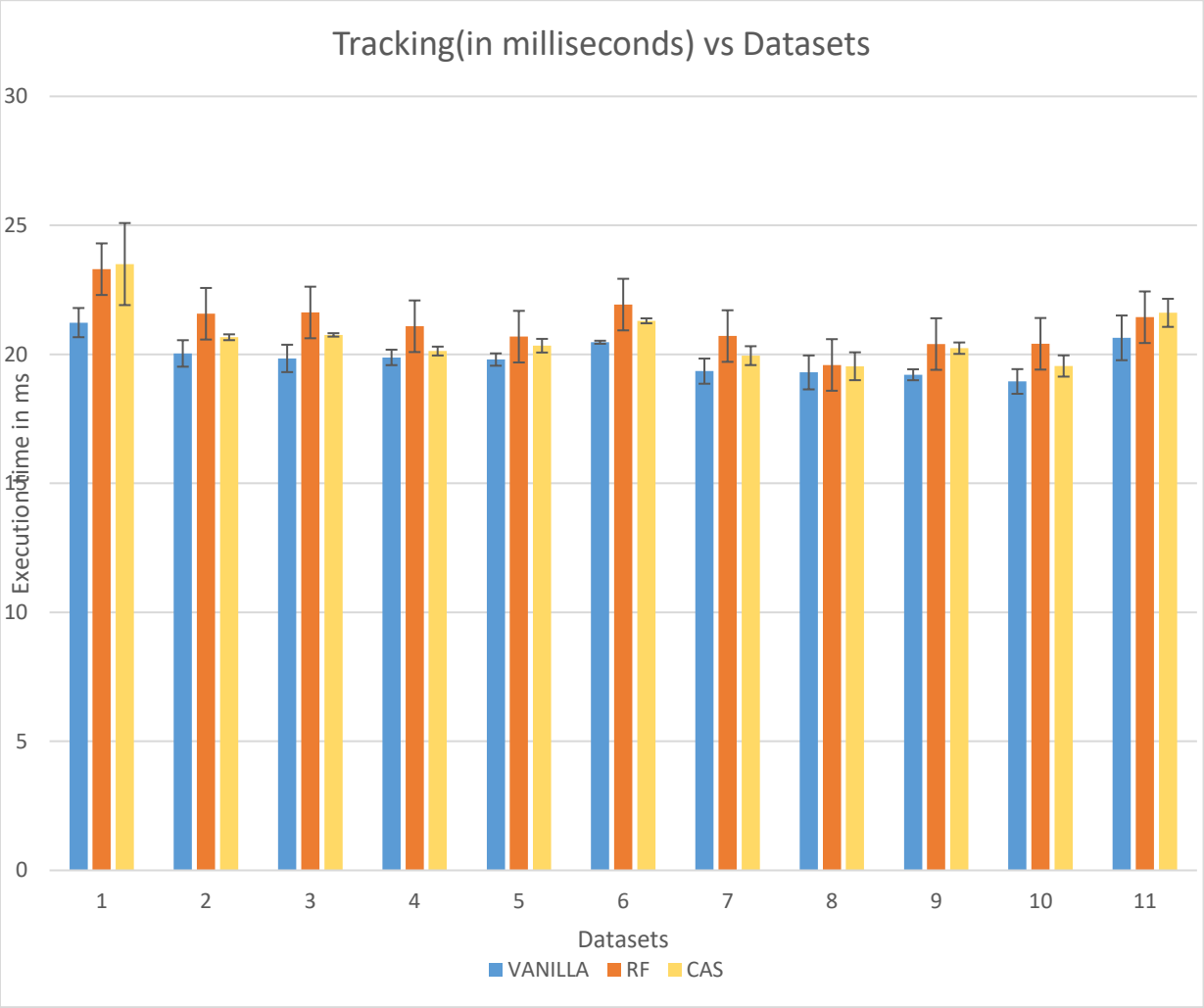
Just like local mapping, this is expected as reference counting will slow down operations due to constant increments and decrements. The CAS implementation is faster than the mutex locked implementation. This is not surprising, as CAS operations will speed up counting when compared to using mutexes.

Figure 4 Tracking Execution Time wrt Datasets.



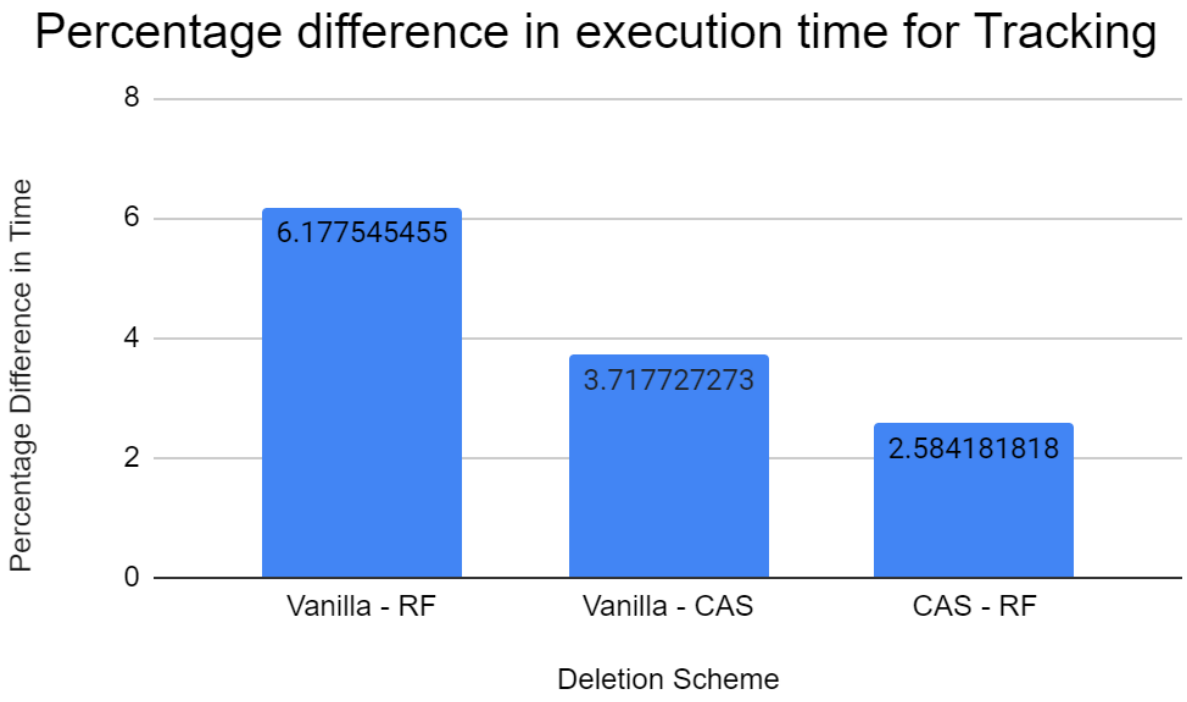
The below figure represents the above figure with error bars as clustered columns.

Figure 5 Tracking Execution Time (with error bars) wrt Datasets.



The figure below shows the percentage difference of total tracking time for the three different versions of ORBSLAM3, the CAS version is closer to plain ORBSLAM3 in terms of execution time. The mutex locked implementation is the slowest, it is about 2% slower than CAS.

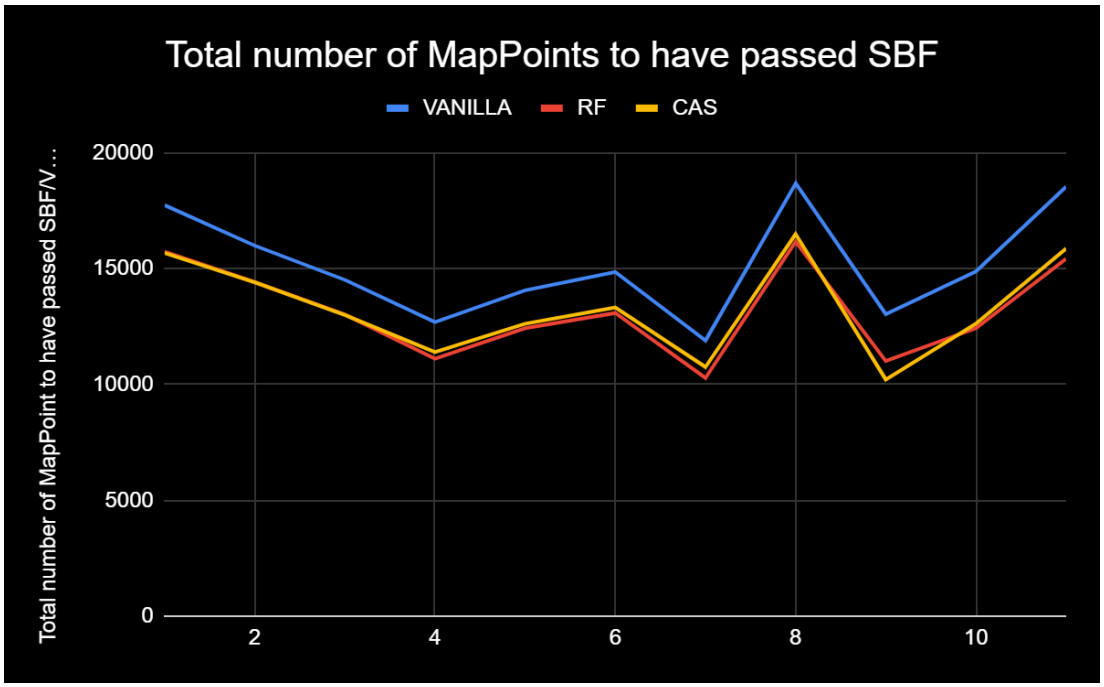
Figure 6 Tracking Percentage Difference in time by Implementation.



The graph below depicts the total number of map points marked bad plotted with respect to the 11 datasets. ORBSLAM3 without deletion is demarcated by the blue line marking the highest number of map points, reference counting implemented with CAS (yellow line) and finally the reference counting implemented with mutexes (red line).

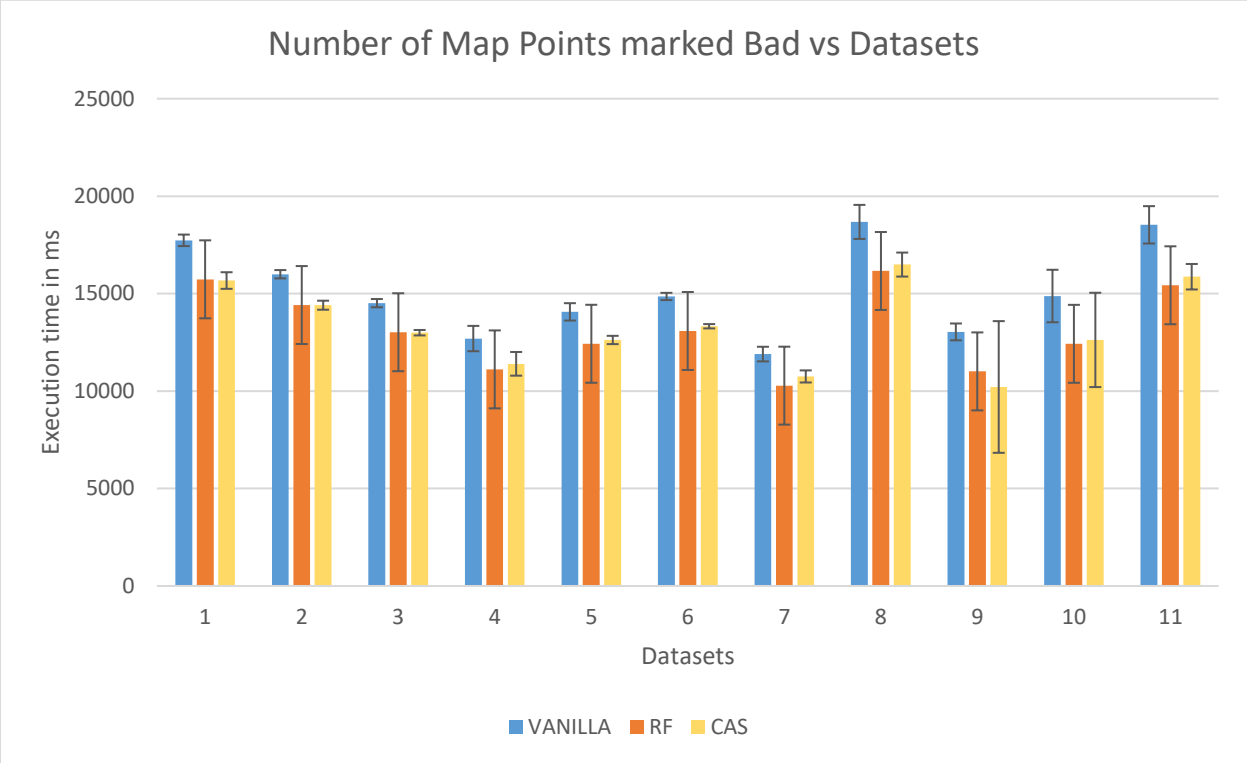
This behavior might be explained by the fact that the system spends a good amount of time to accommodate the reference counts. These computations along with the constraints of processing data in real time might result in a lesser number of map points being marked bad. On some points, CAS performs worse than the mutex locked reference counts, this must be investigated in the future.

Figure 7 Total number of MapPoints to pass SBF wrt Datasets.



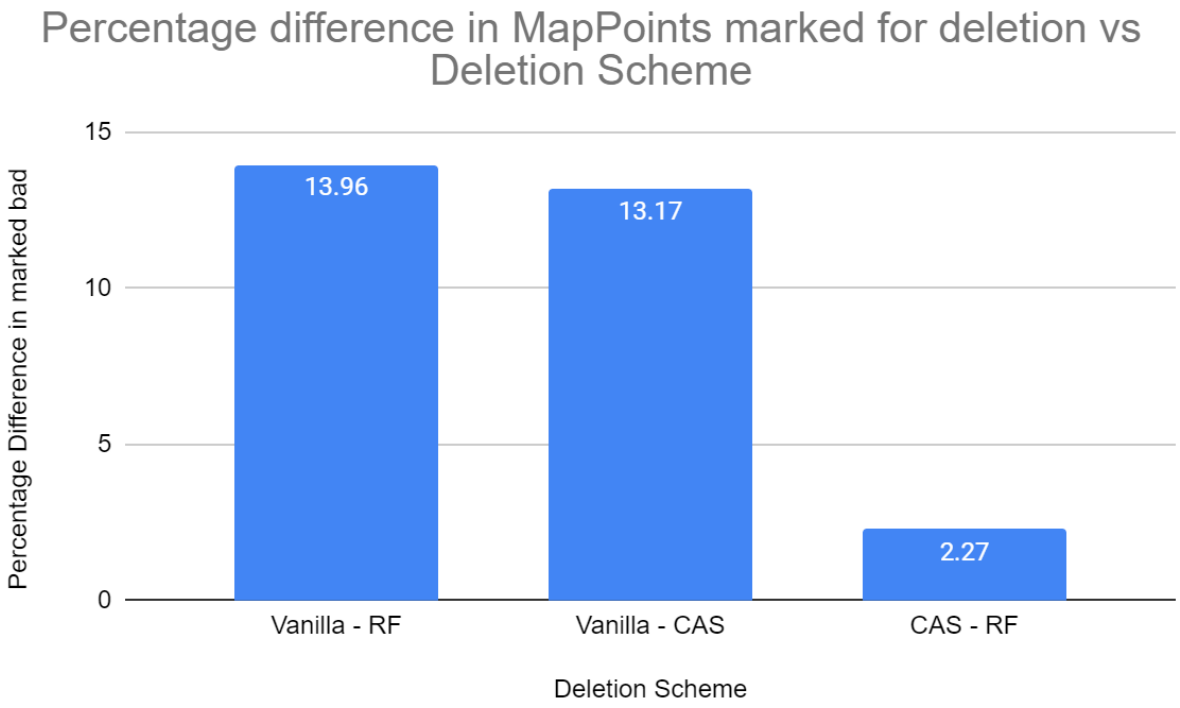
The below figure represents the above figure with error bars as clustered columns.

Figure 8 Total number of MapPoints to pass SBF (with error bars) wrt Datasets.



The figure below shows the percentage difference of map points marked bad for the three different versions of ORBSLAM3, the CAS version is closer to plain ORBSLAM. The mutex locked implementation is the least performant, it marks about 2% less than CAS.

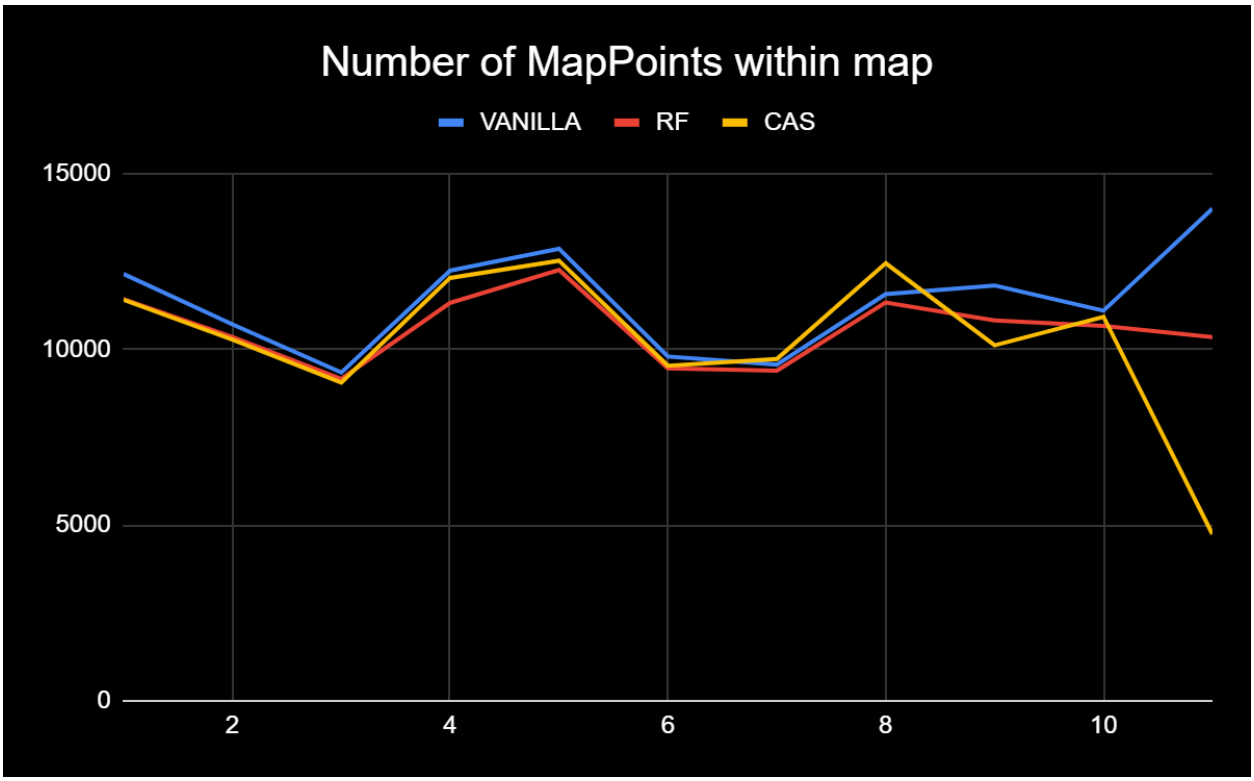
Figure 9 Percentage difference in implementation specific MapPoint marking for deletion.



The graph below depicts the number of map points within the map plotted with respect to the 11 datasets. ORBSLAM3 without deletion demarcated by the blue line performs the best, capturing many map points within the map, followed by the reference counting implemented with CAS (yellow line) and finally reference counting implemented with mutex locks (red line).

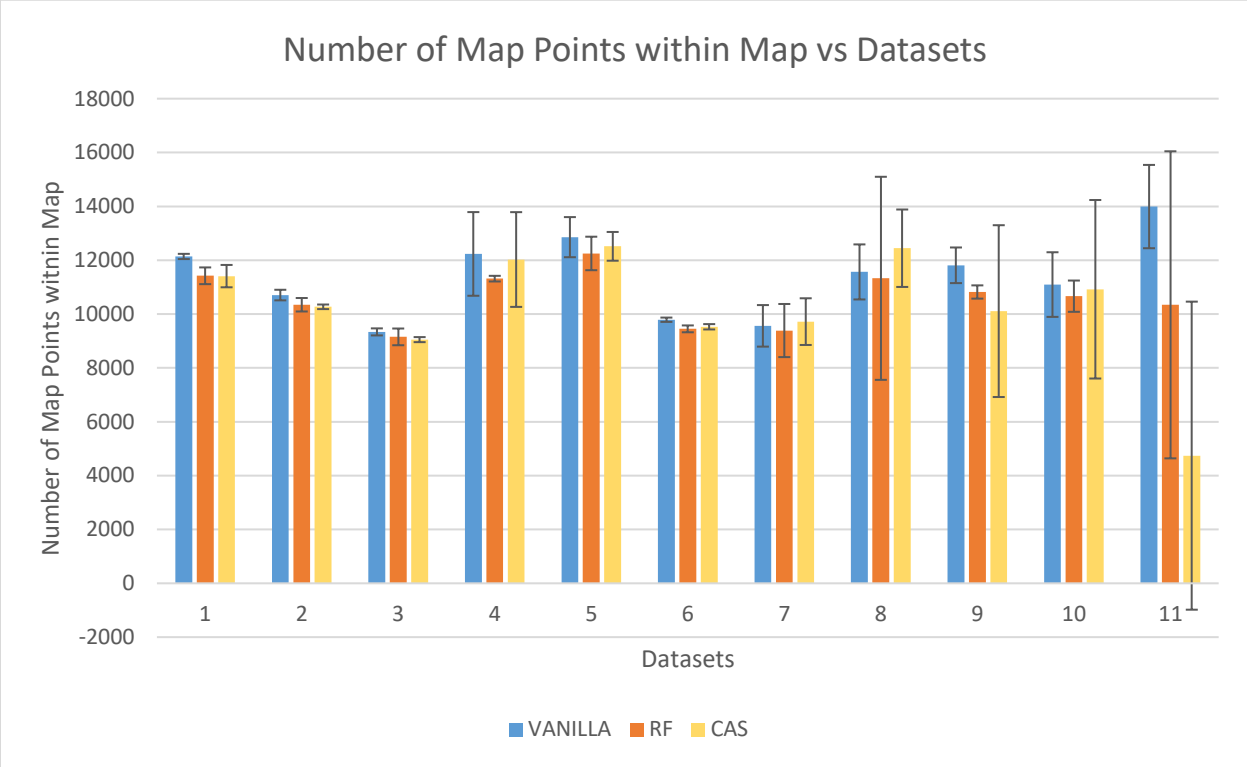
This behavior might be because of the load created by the reference counting infrastructure along with the real time constraints of the system. Also, the yellow line that denotes CAS has several spikes above plain ORBSLAM3 and dips below the mutex implemented count. The cause of this behavior must be investigated in the future.

Figure 10 Implementation specific number of mappoints within map wrt dataset



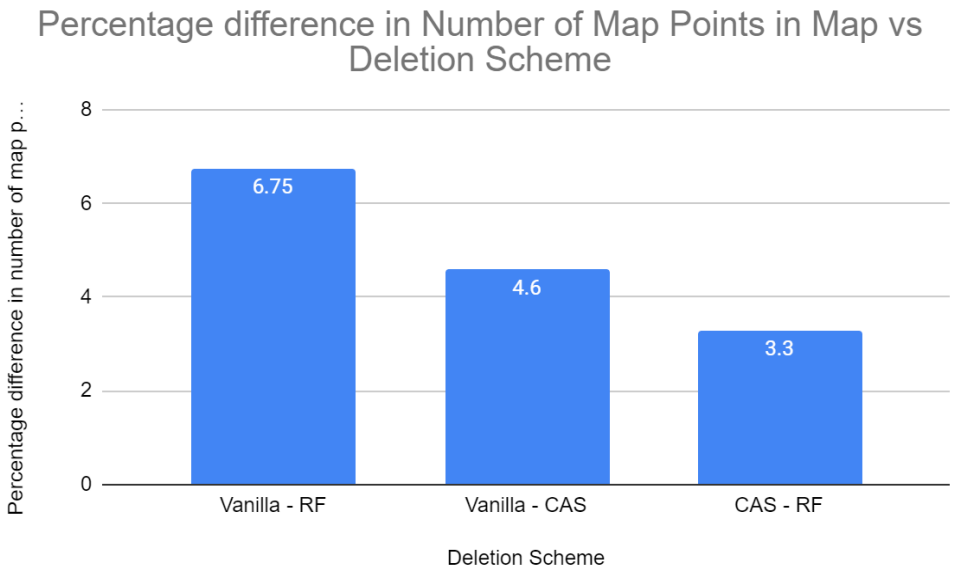
The below figure represents the above figure with error bars as clustered columns.

Figure 11 Implementation specific number of map points within map (with error bars) wrt dataset



The figure below shows the percentage difference of map points in the map for the three different versions of ORBSLAM3, the CAS version is closer to plain ORBSLAM. The mutex locked implementation is the least performant, it collects 3% lesser than CAS. This is in line with expectations.

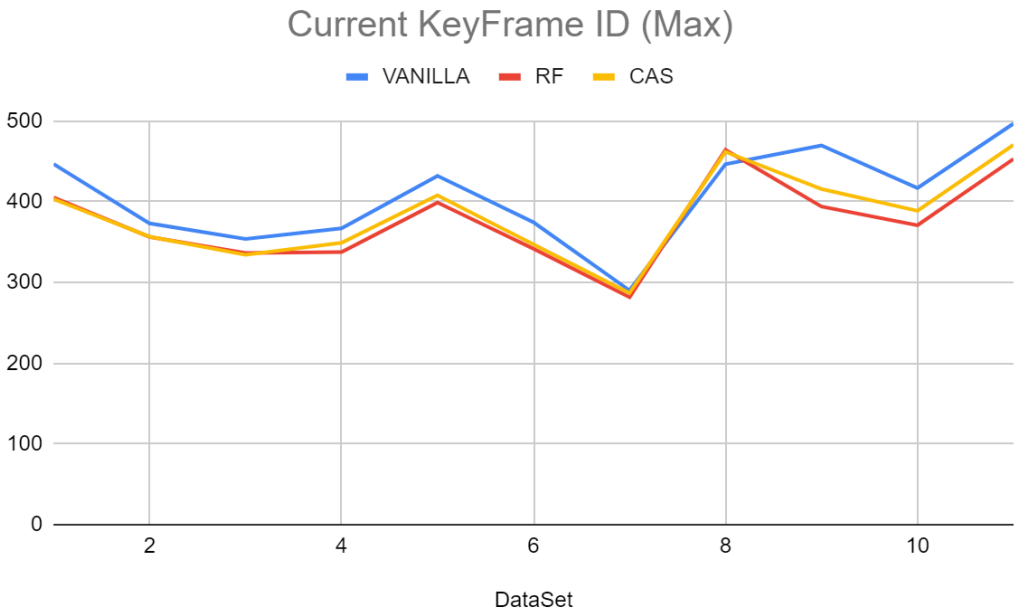
Figure 12 Percentage difference in implementation specific map points within map



The graph below depicts the number of keyframes processed plotted with respect to the 11 datasets. ORBSLAM3 without deletion demarcated by the blue line processes the greatest number of keyframes, followed by the reference counting implemented with CAS (yellow line) and finally reference counting implemented with mutex locks (red line). The explanation for this behavior might be a lower degree of processing keyframes compared to ORBSLAM3 without any deletion.

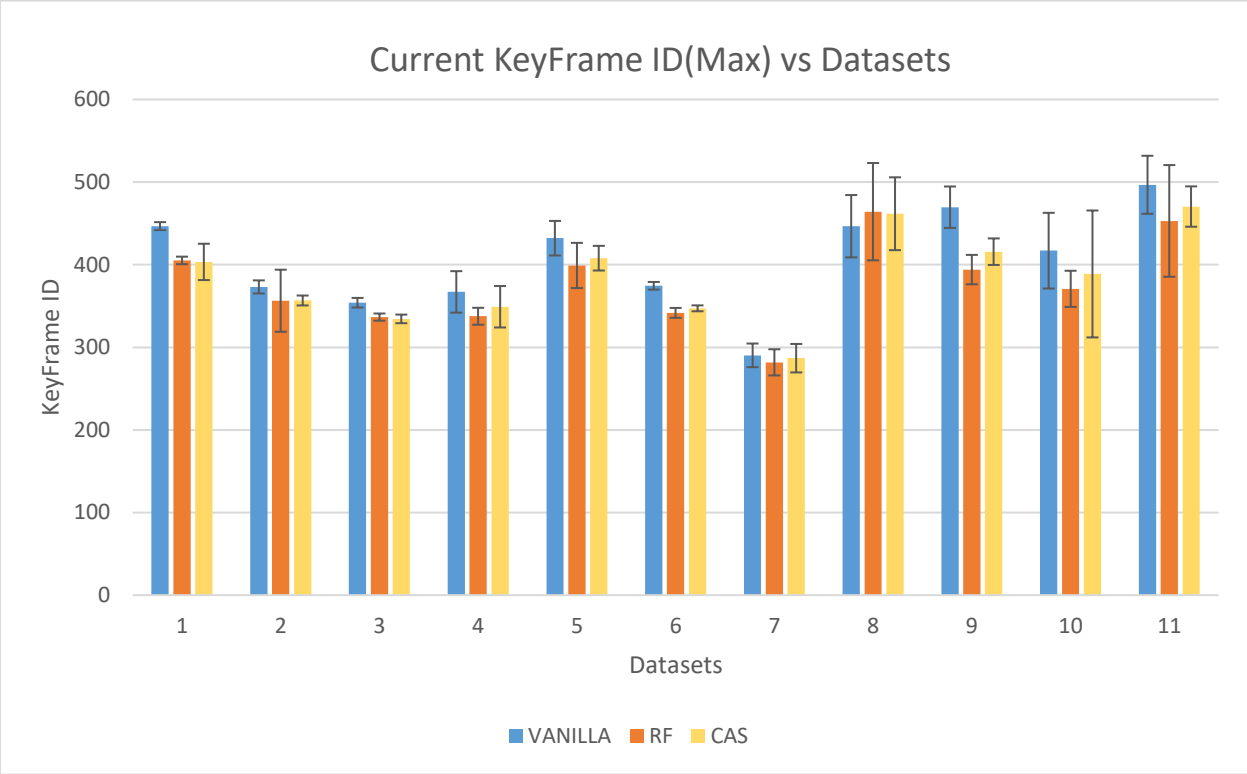
The lower degree of processing might be attributed to the computational load imposed on the system because of the reference counting infrastructure along with real time constraints for processing keyframes and map points. Also, the yellow and red lines have spikes above the blue line. This must be investigated in the future.

Figure 13 KeyFrame ID(Max) wrt Datasets



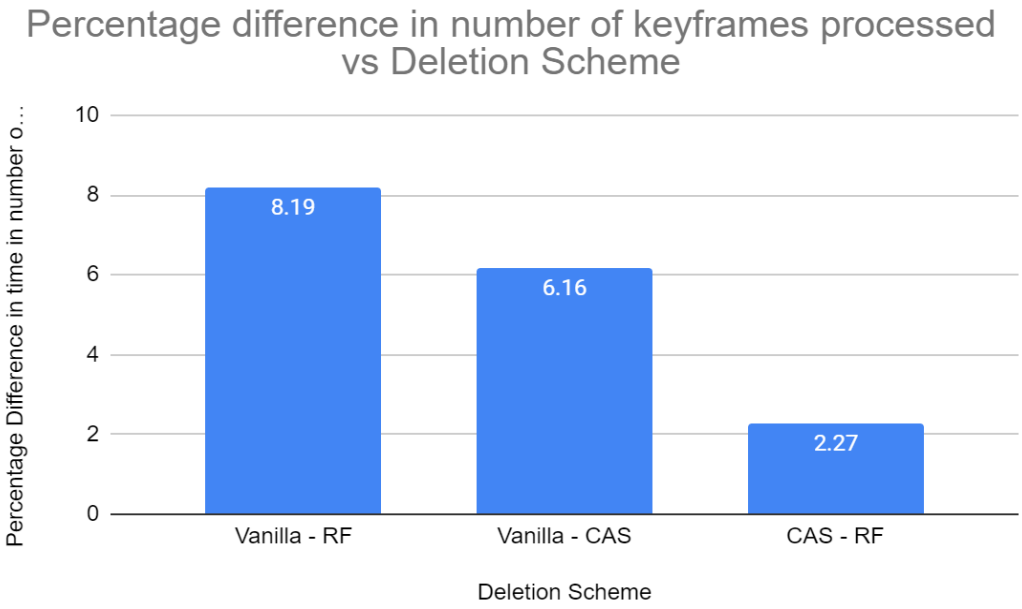
The below figure represents the above figure with error bars as clustered columns.

Figure 14 KeyFrame ID (with Error Bars) wrt Datasets



The figure below shows the percentage difference in the number of processed keyframes for the three different versions of ORBSLAM3, the CAS version is closer to plain ORBSLAM. The mutex locked implementation is the least performant, it processes 2% less than CAS. This is in line with expectations.

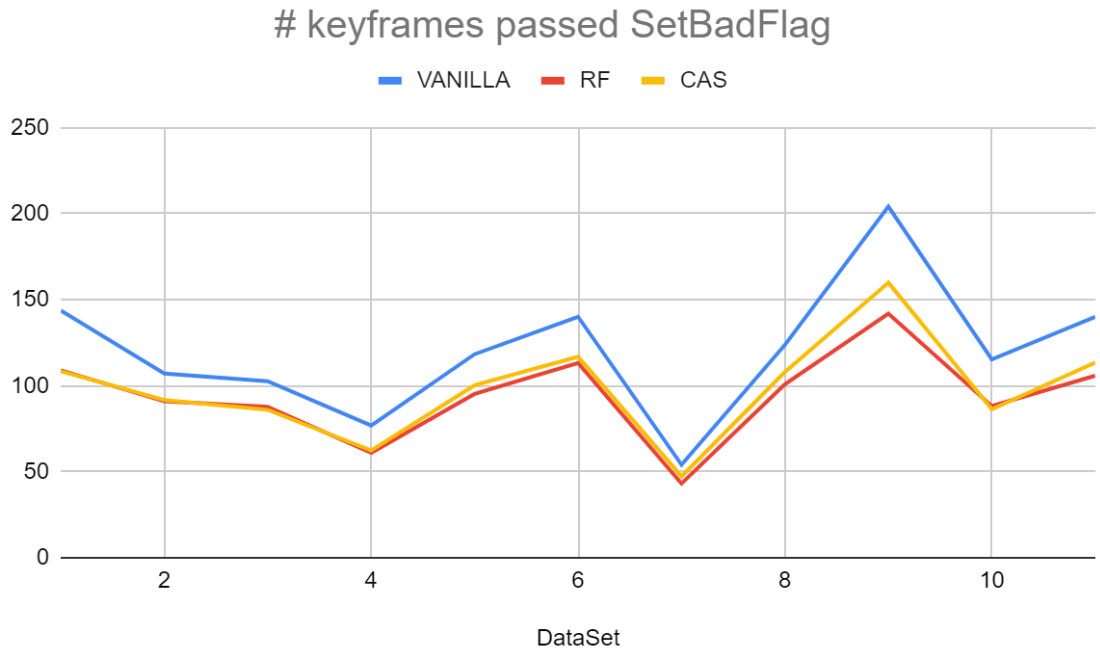
Figure 15 Implementation specific percentage difference for number of processed keyframes



The graph below depicts the number of keyframes marked bad plotted with respect to the 11 datasets. ORBSLAM3 without deletion demarcated by the blue line performs the best marking the highest number, followed by the reference counting implemented with CAS (yellow line) and finally reference counting implemented with mutex locks (red line). This is not surprising as CAS operations will speed up counting when compared to using mutexes.

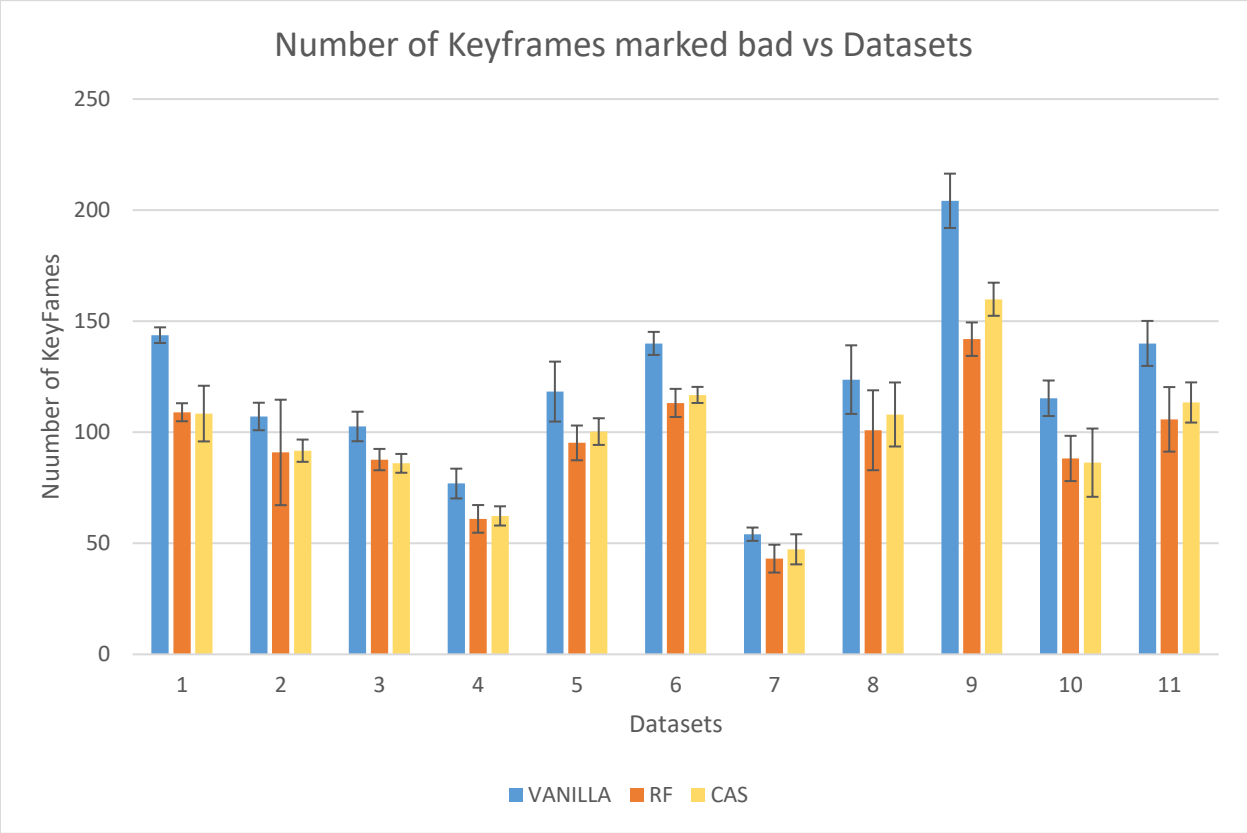
This slowdown in the behavior of the reference counted implementations might be explained by the fact that the system spends a good amount of time to accommodate the reference counts. These computations along with the constraints of processing data in real time might result in a lesser number of keyframes being marked bad.

Figure 16 Implementation specific number of keyframes to pass SBF vs Dataset.



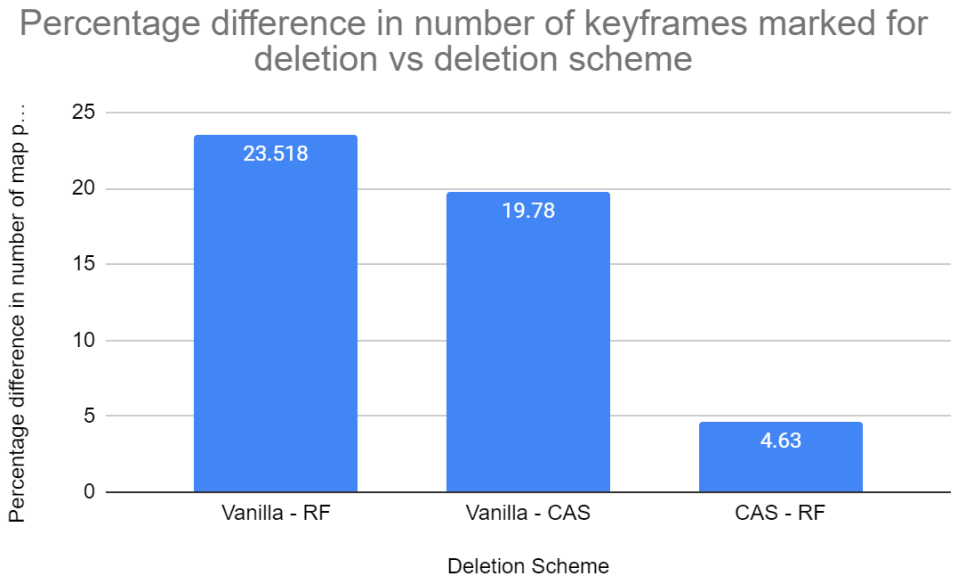
The below figure represents the above figure with error bars as clustered columns.

Figure 17 Implementation specific number of keyframes to pass SBF (with error bars) vs Dataset



The figure below shows the percentage difference in the number of keyframes marked bad for the three different versions of ORBSLAM3, the CAS version is closer to plain ORBSLAM. The mutex locked implementation is the least performant, it marks 4% lesser than CAS. This is in line with expectations.

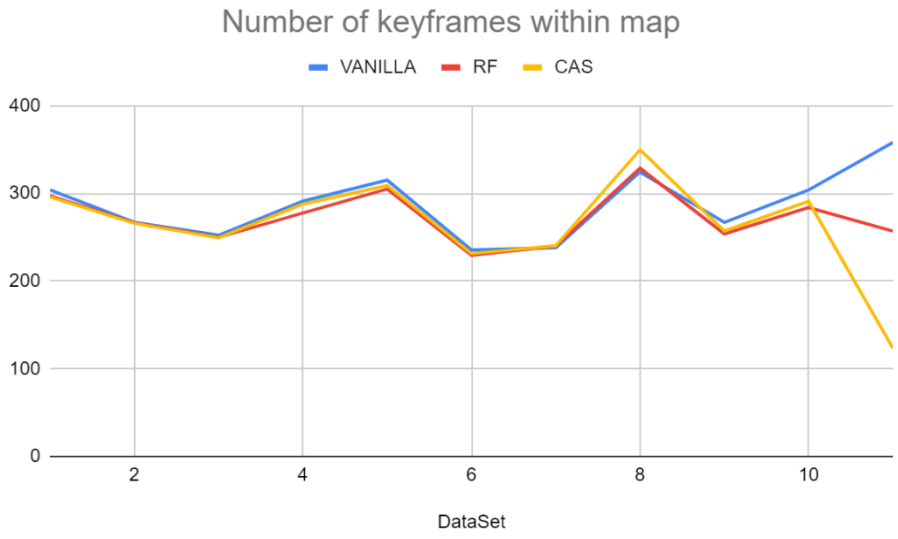
Figure 18 Implementation specific percentage difference for keyframes marked for deletion



The graph below depicts the number of keyframes within the map plotted with respect to the 11 datasets, ORBSLAM3 without deletion demarcated by the blue line performs the best, capturing many keyframes within the map, followed by the reference counting implemented with CAS (yellow line) and finally reference counting implemented with mutex locks (red line).

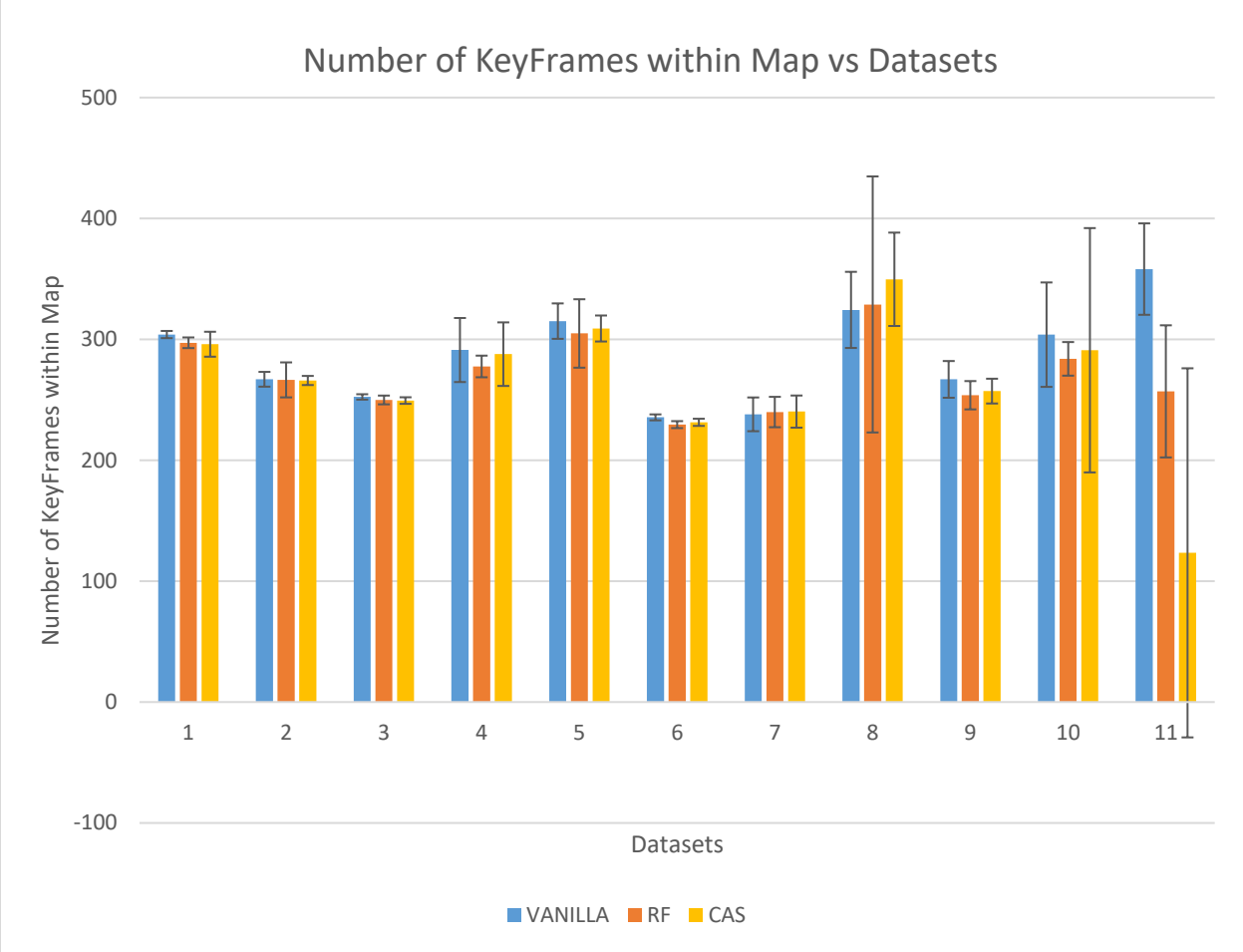
This behavior might be because of the load created by the reference counting infrastructure along with the real time constraints of the system affects the addition of keyframes to the map. However, the yellow line has several spikes above plain ORBSLAM3 and dips below the mutex implemented count. This must be investigated in the future.

Figure 19 Implementation specific number of keyframes in map vs Dataset



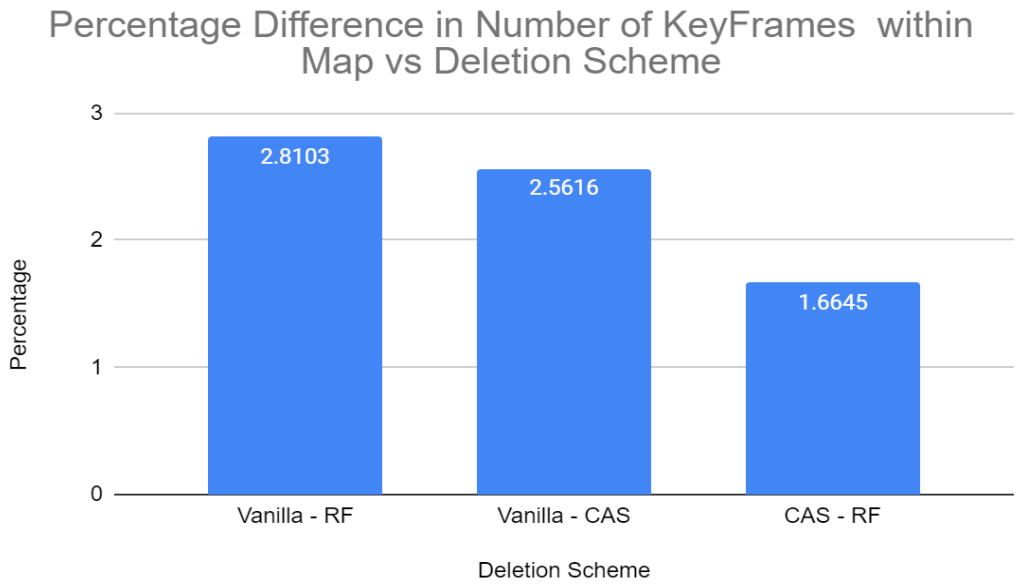
The below figure represents the above figure with error bars as clustered columns.

Figure 20 19 Implementation specific number of keyframes in map (with error bars) vs Dataset



The figure below shows the percentage difference in the number of keyframes within the map for the three different versions of ORBSLAM3, the CAS version is closer to plain ORBSLAM. The mutex locked implementation is the least performant, it is about 1% less than CAS. This is in line with expectations.

Figure 21 Implementation specific percentage difference of keyframes within map



The figures below show the number of keyframes deleted. It is in line with the general trend seen above; CAS outperforms reference counting with mutexes.

Figure 22 Implementation specific number of keyframes deleted.

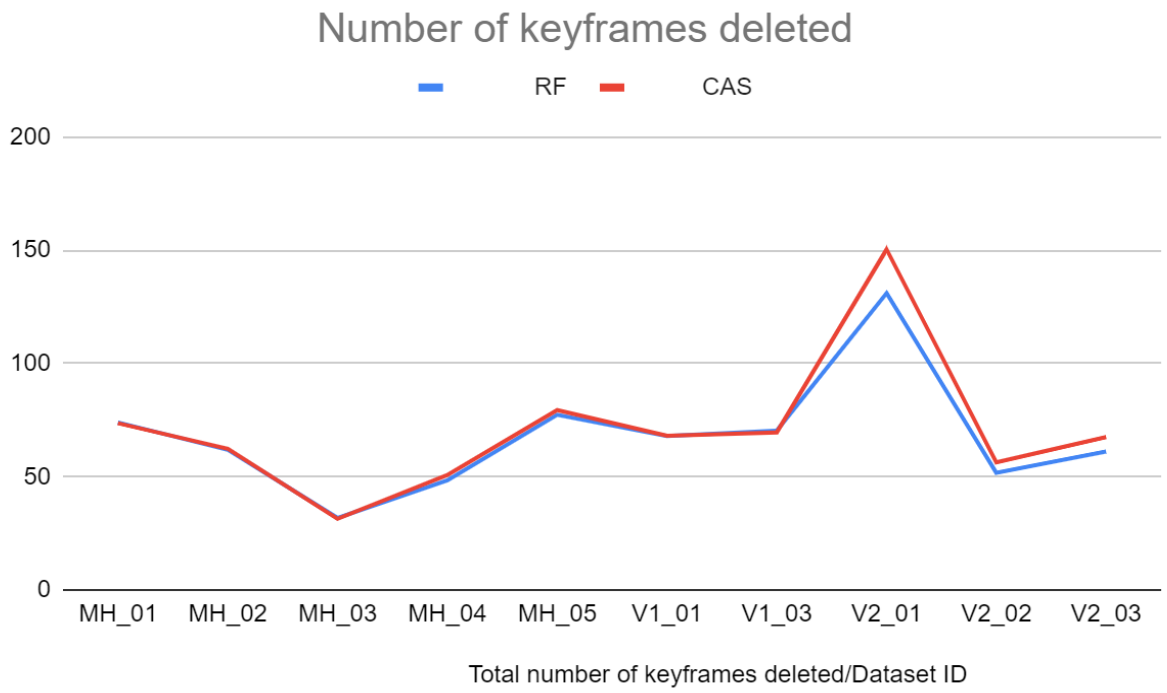
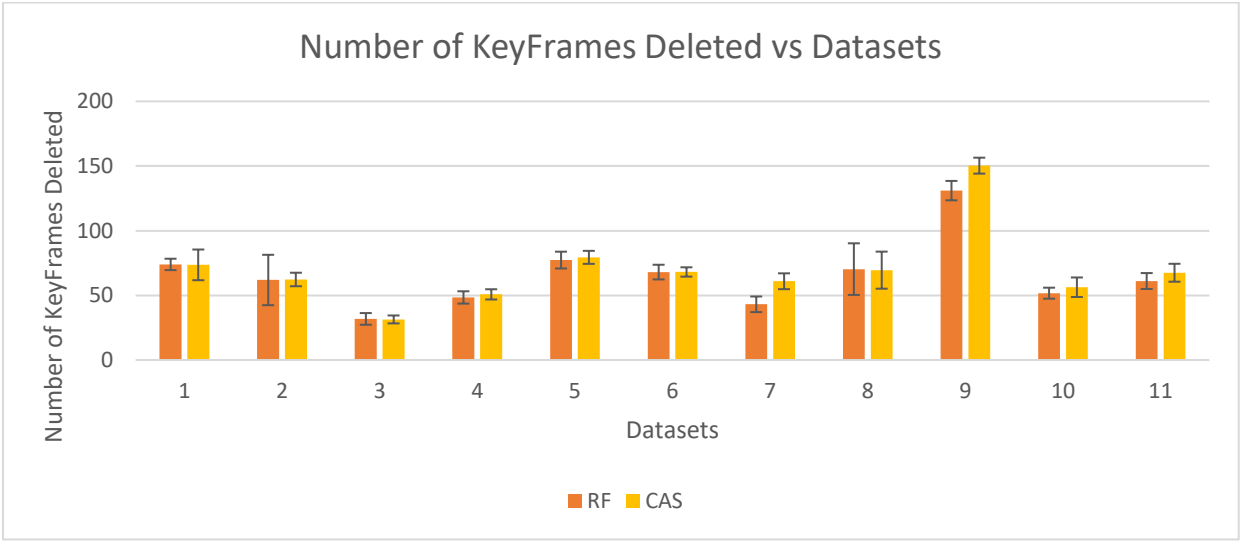


Figure 23 Implementation specific number of keyframes deleted (with error bars)



The figures below show the number of map points deleted. It is also in line with the general trend seen above; CAS outperforms reference counting with mutexes. The performance exhibited by CAS falls below performance seen from the reference count implemented with mutexes for the dataset named V2_01. This must be investigated in the future.

Figure 24 Implementation specific number of map points deleted.

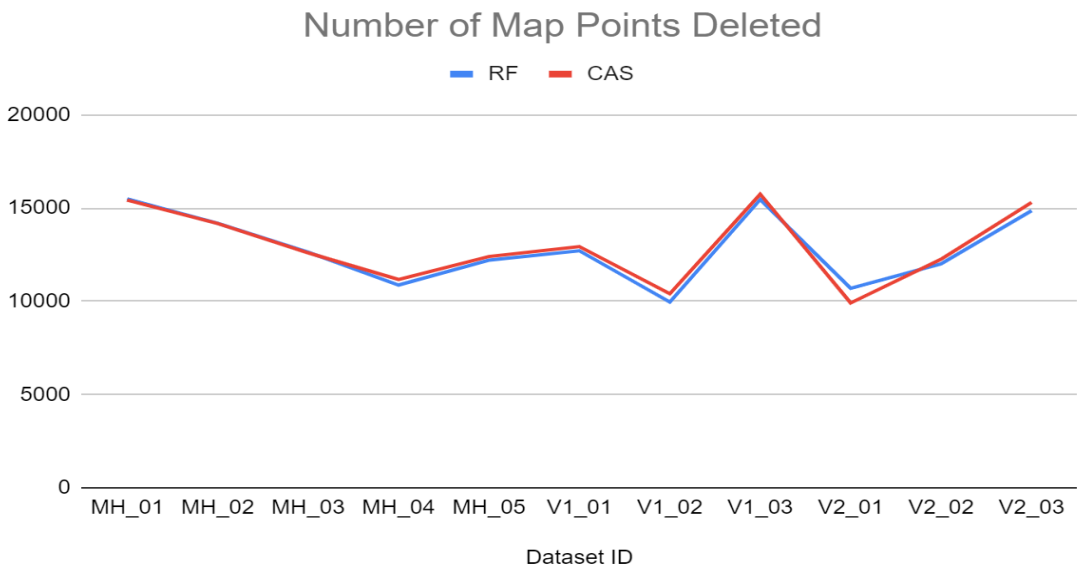
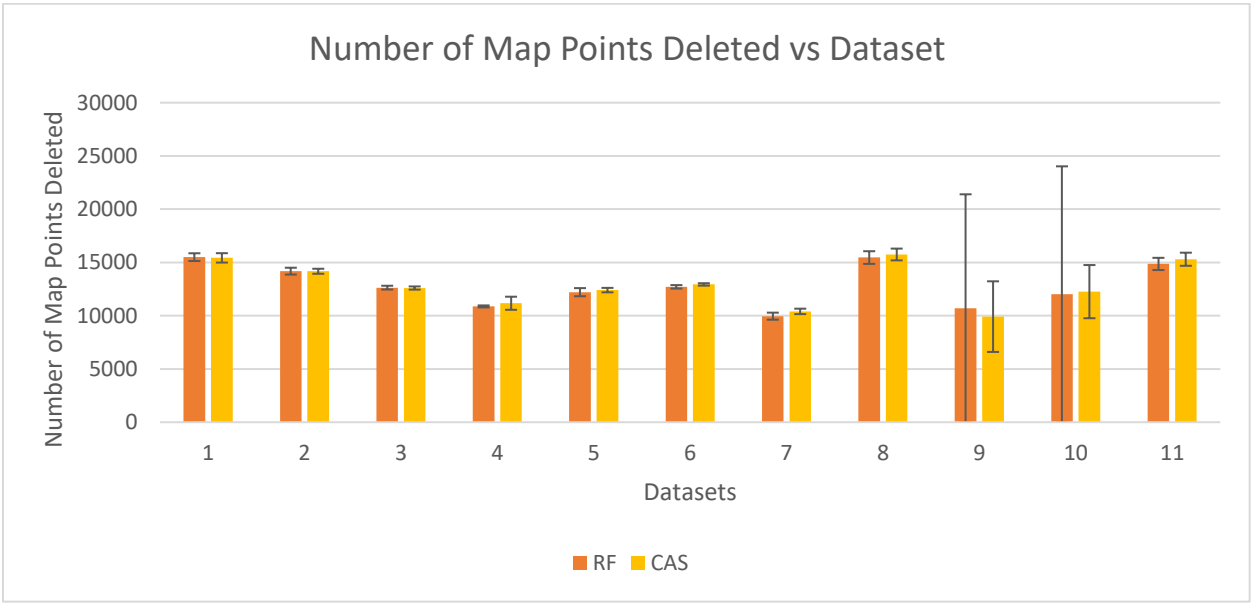


Figure 25 Implementation specific number of map points deleted (with error bars).



To conclude, CAS exhibits slightly better performance in terms of speed compared to reference counting implemented using mutexes. This can be surmised from the observations made from the graphs above.

ORBSLAM3 Deletion Statistics with Thread Based Reference Counting.

ORBSLAM3 was profiled on the EuRoC dataset with an Intel i7-11857G with 32 gigabytes of memory. The EuRoC dataset consists of 11 sequences namely – Machine Hall 01, Machine Hall 02, Machine Hall 03, Machine Hall 04, Machine Hall 05, Vicon Room 1 01, Vicon Room 1 02, Vicon Room 1 03, Vicon Room 2 01, Vicon Room 2 02, Vicon Room 2 03. The times for measurements pertaining to function execution times are in milliseconds. The results are discussed at the end of this section.

The calculations for all quantities are averaged across 5 iterations of each sequence. The following tables represent the number of keyframes, and map points processed and marked for deletion. It also shows the number of deleted keyframes and map points along with related statistics. Execution times of the local mapping, loop closing, and the tracking thread are also tabulated below. These statistics were captured to evaluate both the reference counting implementations.

The following consists of some abbreviated terms that are used in the tables below.

(SBF expands to SetBadFlag)

(kfs/KFs expands to KeyFrames) and (MP expands to MapPoint)

(LM expands to Local Mapping)

(LBA expands to Local Bundle Adjustment)

(No. expands to Number)

MH_01_TRC

Table 163 MH_01_TRC-General Statistics

#Iterations	Total number of keyframes to have passed the SetBadFlag	Total No. of deleted keyframes	Number of keyframes within map	Percentage of deletion
1	89	61	279	68.54%
2	89	62	275	69.66%
3	99	70	276	70.71%
4	89	61	276	68.54%
5	83	53	276	63.86%
	89.8	61.4	276.4	68.26%

Table 164 MH_01_TRC-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion
1	14596	14402	10733	98.67%
2	14572	14399	10741	98.81%
3	14889	14699	10769	98.72%
4	14859	14693	10698	98.88%
5	14774	14594	10808	98.78%
	14738	14557.4	10749.8	98.77%

Table 165 MH_01_TRC-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	8.3515	14.144	45.504	312.29	33.73	412.1
2	8.0982	13.809	45.176	308.96	32.75	407.01
3	8.3385	13.41	45.042	303.21	31.494	399.8
4	8.1262	13.369	45.032	306.06	33.312	404.1
5	8.3696	14.063	45.764	317.27	32.988	416.55
	8.2568	13.759	45.3036	309.558	32.8548	407.91

Table 166 MH_01_TRC-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	12.015	2.2625	13.187	0.10291	30.15
2	11.802	2.2114	12.618	0.12793	29.336
3	11.711	2.214	12.663	0.10381	29.259
4	11.785	2.2073	12.552	0.097773	29.194
5	11.907	2.2192	12.773	0.11832	29.594
	11.844	2.22288	12.7586	0.110148	29.506

MH_02_TRC

Table 167 MH_02_TRC-General Statistics

#Iterations	Total number of keyframes to have passed the SetBadFlag	Total No. of deleted keyframes	Number of keyframes within map	Percentage of deletion
1	73	50	245	68.49%
2	69	46	251	66.67%
3	71	47	236	66.20%
4	69	45	258	65.22%
5	74	52	243	70.27%
	71.2	48	246.6	67.37%

Table 168 MH_02_TRC-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion
1	12836	12678	9665	98.77%
2	13287	13177	9720	99.17%
3	13409	13252	9406	98.83%
4	13752	13603	9745	98.92%
5	13293	13135	9372	98.81%
	13315.4	13169	9581.6	98.90%

Table 169 MH_02_TRC-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	7.4749	13.679	42.211	293.98	27.878	383.29
2	7.3558	13.038	41.955	308.8	28.437	397.57
3	7.6124	13.876	43.276	318.65	31.839	412.99
4	7.2525	12.86	41.392	299.98	27.191	386.76
5	7.1392	12.68	41.154	297.82	26.424	383.18
	7.36696	13.2266	41.9976	303.846	28.3538	392.758

Table 170 MH_02_TRC-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	11.634	2.0732	11.036	0.09041	27.122
2	11.806	2.1787	9.6697	0.095661	26.069
3	11.574	2.1418	10.512	0.096935	26.62
4	11.738	2.1066	9.3751	0.099095	25.606
5	11.506	2.0709	9.3139	0.086789	25.197
	11.6516	2.11424	9.98134	0.093778	26.1228

MH_03_TRC

Table MH_03_TRC-General Statistics

#Iterations	Total number of keyframes to have passed the SetBadFlag	Total No. of deleted keyframes	Number of keyframes within map	Percentage of deletion
1	77	31	233	40.26%
2	71	26	227	36.62%
3	78	34	231	43.59%
4	77	37	227	48.05%
5	76	27	227	35.53%
	75.8	31	229	40.81%

Table 171 MH_03_TRC-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion
1	12478	12366	8346	99.10%
2	12150	12035	8231	99.05%
3	12476	12337	8344	98.89%
4	12143	12040	8294	99.15%
5	12173	12059	8381	99.06%
	12284	12167.4	8319.2	99.05%

Table 172 MH_03_TRC-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	7.2462	12.5	36.564	256.79	29.797	341.15
2	7.1575	13.03	37.181	275.64	30.795	361.85
3	7.3126	12.137	36.195	263.28	29.451	346.58
4	7.1736	12.27	36.103	251.45	28.995	334.25
5	7.2581	12.497	35.877	265.82	29.704	349.31
	7.2296	12.4868	36.384	262.596	29.7484	346.62

Table 173 MH_03_TRC-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	11.355	2.0893	8.7415	0.10274	24.573
2	11.49	2.1308	8.6096	0.088671	24.593
3	11.331	2.0977	9.3483	0.081249	25.121
4	11.344	2.1094	8.4966	0.081742	24.251
5	11.44	2.1201	8.7816	0.082285	24.732
	11.392	2.10946	8.79552	0.0873374	24.654

MH_04_TRC

Table 174 MH_04_TRC-General Statistics

#Iterations	Total number of keyframes to have passed the SetBadFlag	Total No. of deleted keyframes	Number of keyframes within map	Percentage of deletion
1	66	58	277	87.88%
2	58	50	255	86.21%
3	50	43	255	86%
4	45	39	254	86.67%
5	52	44	274	84.62%
	54.2	46.8	263	86.27%

Table 175 MH_04_TRC-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion
1	11092	10807	10856	97.43%
2	10257	10166	10451	99.11%
3	10628	10523	10363	99.01%
4	10252	10173	10358	99.23%
5	10996	10703	11766	97.34%
	10645	10474.4	10758.8	98.42%

Table 176 MH_04_TRC-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	5.7257	14.493	27.718	144.18	19.993	210.25
2	5.9774	13.551	30.006	171.46	22.434	242.26
3	5.8084	13.728	29.539	171.28	23.339	242.49
4	5.7227	13.781	30.11	171.6	22.506	242.5
5	6.0229	15.248	28.92	158.03	23.011	230.19
	5.85142	14.1602	29.2586	163.31	22.2566	233.538

Table 177 MH_04_TRC-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	10.116	2.0641	7.0112	0.092378	21.317
2	10.223	2.0629	7.7895	0.088984	22.176
3	9.9423	1.9497	7.3957	0.087822	21.316
4	10.238	2.0768	7.8686	0.091753	22.266
5	10.474	2.0898	7.7526	0.082815	22.536
	10.19866	2.04866	7.56352	0.0887504	21.9222

MH_05_TRC

Table 178 MH_05_TRC-General Statistics

#Iterations	Total number of keyframes to have passed the SetBadFlag	Total No. of deleted keyframes	Number of keyframes within map	Percentage of deletion
1	80	66	290	82.50%
2	79	63	271	79.75%
3	66	62	306	93.94%
4	86	74	307	86.05%
5	82	69	291	84.15%
	78.6	66.8	293	85.28%

Table 179 MH_05_TRC-MapPoint Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion
1	11736	11536	11730	98.30%
2	11560	11400	10747	98.62%
3	12292	12097	12450	98.41%
4	12314	11925	12187	96.84%
5	11873	11647	11660	98.10%
	11955	11721	11754.8	98.05%

Table 180 MH_05_TRC-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	6.4069	13.741	31.899	174.7	22.178	247.26
2	6.4219	13.754	32.911	182.81	27.052	261.82
3	6.0115	13.892	32.149	163.85	20.905	235.77
4	6.2428	14.301	31.115	163.33	20.215	234.28
5	6.2414	13.794	31.035	168.47	21.544	240.08
	6.2649	13.8964	31.8218	170.632	22.3788	243.842

Table 181 MH_05_TRC-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	10.865	2.2832	9.2201	0.10842	24.875
2	10.504	2.1554	8.5192	0.10327	23.536
3	10.864	2.1592	10.059	0.10593	25.509
4	10.84	2.2455	8.3748	0.10879	23.942
5	10.753	2.2983	9.031	0.094234	24.602
	10.7652	2.22832	9.04082	0.1041288	24.4928

V1_01_TRC

Table 182 V1_01_TRC-General Statistics

#Iterations	Total number of keyframes to have passed the SetBadFlag	Total No. of deleted keyframes	Number of keyframes within map	Percentage of deletion
1	82	46	214	56.10%
2	83	43	213	51.81%
3	139	98	225	70.50%
4	83	43	215	51.81%
5	85	47	210	55.29%
	94.4	55.4	215.4	57.10%

Table 183 V1_01_TRC-MapPoint Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion
1	12005	11908	8858	99.19%
2	12090	11999	8800	99.25%
3	13337	13162	9507	98.69%
4	12176	12062	8874	99.06%
5	11877	11773	8876	99.12%
	12297	12180.8	8983	99.06%

Table 184 V1_01_TRC-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	8.6803	18.207	41.359	334.3	43.87	444.02
2	8.7265	18.32	41.03	325.81	42.264	433.81
3	7.0579	10.737	31.012	270.13	33.036	350.4
4	8.5123	17.4	41.361	330.91	42.559	438.38
5	8.1785	17.676	40.207	321.97	41.654	427.37
	8.2311	16.468	38.9938	316.624	40.6766	418.796

Table 185 V1_01_TRC-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	10.929	2.5395	12.011	0.11145	27.992
2	10.847	2.559	11.903	0.12223	27.805
3	8.8568	2.1271	11.567	0.098581	24.875
4	10.852	2.555	12.354	0.09894	28.213
5	10.633	2.4877	11.924	0.10097	27.48
	10.42356	2.45366	11.9518	0.1064342	27.273

V1_02_TRC

Table 186 V1_02_TRC-General Statistics

#Iterations	Total number of keyframes to have passed the SetBadFlag	Total No. of deleted keyframes	Number of keyframes within map	Percentage of deletion
1	26	8	214	30.77%
2	31	9	207	29.03%
3	39	21	236	53.85%
4	32	6	204	18.75%
5	38	15	240	39.47%
	33.2	11.8	220.2	34.37%

Table 187 V1_02_TRC-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion
1	9343	9266	8422	99.18%
2	9823	9690	7870	98.65%
3	10101	9823	9786	97.25%
4	9515	9367	8112	98.44%
5	10296	10154	9795	98.62%
	9815.6	9660	8797	98.43%

Table 188 V1_02_TRC-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	5.7775	16.779	29.35	209.42	19.43	278.93
2	6.2202	17.844	30.483	211.91	19.277	283.86
3	5.6258	17.985	25.645	174.1	15.699	236.4
4	6.014	17.59	29.922	213.57	20.805	285.01
5	5.946	17.96	26.252	179.2	15.968	240.49
	5.9167	17.6316	28.3304	197.64	18.2358	264.938

Table 189 V1_02_TRC-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	10.356	2.269	7.5188	0.081439	22.55
2	10.209	2.3472	7.4155	0.096776	22.293
3	10.286	2.3633	9.5838	0.09595	24.84
4	10.264	2.2573	7.986	0.0728	22.82
5	10.311	2.3343	7.5432	0.11347	22.61
	10.2852	2.31422	8.00946	0.092087	23.0226

V1_03_TRC

Table 190 V1_03_TRC-General Statistics

#Iterations	Total number of keyframes to have passed the SetBadFlag	Total No. of deleted keyframes	Number of keyframes within map	Percentage of deletion
1	64	35	277	54.69%
2	75	42	288	56%
3	69	39	276	56.52%
4	84	43	294	51.19%
5	92	65	46	70.65%
	76.8	44.8	236.2	57.81%

Table 191 V1_03_TRC-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion
1	14577	14389	10126	98.71%
2	15160	14938	10027	98.54%
3	14262	14042	9795	98.46%
4	14651	14317	10215	97.72%
5	14792	14384	1384	97.24%
	14688.4	14414	8309.4	98.13%

Table 192 V1_03_TRC-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	6.2158	17.941	29.379	170.26	20.312	243.05
2	5.7808	17.387	27.864	145.43	18.59	212.44
3	6.0302	18.515	29.099	165.42	18.841	236.89
4	5.5583	17.052	27.07	153.37	17.453	219.65
5	5.1815	16.908	22.779	115.27	13.894	171.37
	5.75332	17.5606	27.2382	149.95	17.818	216.68

Table 193 V1_03_TRC-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	9.663	2.242	7.7351	0.109	22.212
2	9.44	2.204	6.474	0.0946	20.437
3	10.008	2.274	7.744	0.0979	22.649
4	9.8493	2.1893	7.9781	0.10231	22.634
5	9.6256	2.1825	5.56	0.08385	20.052
	9.71718	2.21836	7.09824	0.097532	21.5968

V2_01_TRC

Table 194 V2_01_TRC-General Statistics

#Iterations	Total number of keyframes to have passed the SetBadFlag	Total No. of deleted keyframes	Number of keyframes within map	Percentage of deletion
1	115	197	232	93.04
2	127	116	227	91.33
3	125	116	229	92.79
4	116	108	233	93.1
5	123	115	256	93.49
	121.2	130.4	235.4	92.75

Table 195 V2_01_TRC-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion
1	10278	10091	10387	98.18%
2	10181	9907	10044	97.31%
3	10346	10044	10229	97.08%
4	10341	10158	10491	98.23%
5	10707	10463	10340	97.72%
	10370.6	10132.6	10298.2	97.70%

Table 196 V2_01_TRC-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	7.6597	20.33	33.342	185.19$	28.768	274.15
2	7.303	18.771	32.322	182.37	27.84	266.91
3	7.72	20.041	32.804	180.47	29.114	268.46
4	7.4011	19.806	32.683	182.78	28.779	270.32
5	6.9459	19.173	31.025	167.73	24.749	248.17
	7.40594	19.6242	32.4352	179.71$	27.85	265.602

Table 197 V2_01_TRC-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	10.897	2.7997	8.3	0.11032	24.155
2	10.699	2.7467	8.6241	0.10077	24.224
3	10.696	2.7484	10.026	0.11755	25.596
4	10.658	2.7775	8.2767	0.10331	23.834
5	10.725	2.6708	7.7187	0.10326	23.237
	10.735	2.74862	8.5891	0.107042	24.2092

V2_02_TRC

Table 198 V2_02_TRC-General Statistics

#Iterations	Total number of keyframes to have passed the SetBadFlag	Total No. of deleted keyframes	Number of keyframes within map	Percentage of deletion
1	85	49	354	57.64
2	64	32	255	50
3	62	37	269	59.67
4	89	53	265	59.55
5	64	32	268	50
	72.8	40.6	282.2	55.372

Table 199 V2_02_TRC-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion
1	14611	14354	12737	98.24
2	11914	11796	9186	99
3	11008	10854	10330	98.6
4	11523	11352	9880	98.51
5	11758	11638	9662	98.9
	12162.8	11998.8	10359	98.65

Table 200 V2_02_TRC-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	6.1826	17.838	30.07	149.46	20.562	222.21
2	7.0968	18.63	33.33	225.6	29.187	312.35
3	6.4256	16.697	30.524	216.95	24.507	292.98
4	6.8292	16.541	31.021	202.83	26.09	282.09
5	6.7642	17.965	31.71	221.17	25.149	301.35
	6.65968	17.5342	31.331	203.202	25.099	282.196

Table 201 V2_02_TRC-MapPoint-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	10.73	2.1301	9.9498	0.11996	25.206
2	10.493	2.1699	9.9439	0.0963	25.043
3	10.343	2.1909	11.424	0.13378	26.506
4	10.31	2.2147	11.287	0.11855	26.217
5	10.452	2.2306	10.684	0.10643	25.822
	10.4656	2.18724	10.65774	0.115004	25.7588

V2_03_TRC

Table 202 V2_03_TRC-General Statistics

#Iterations	Total number of keyframes to have passed the SetBadFlag	Total No. of deleted keyframes	Number of keyframes within map	Percentage of deletion
1	120	74	340	61.66
2	98	47	297	47.95
3	80	46	367	57.5
4	84	48	362	57.14
5	80	51	336	63.75
	92.4	53.2	340.4	57.6

Table 203 V2_03_TRC-MapPoint-Deletion Statistics

#Iterations	Total number of MapPoints to have passed SBF	Total number of deleted MapPoints	Number of MapPoints within map	Percentage of deletion
1	14592	14331	12937	98.21
2	14282	13983	11578	97.9
3	15292	15047	14326	98.39
4	15369	15088	14531	98.17
5	14695	14389	12881	97.91
	14846	14567.6	13250.6	98.116

Table 204 V2_03_TRC-LocalMapping Statistics

#Iterations	KF Insertion	MP Culling	MP Creation	LBA	KF Culling	Total Local Mapping
1	4.5951	10.146	19.229	109.99	16.454	159.11
2	5.5982	18.16	24.48	132.06	18.65	196.76
3	5.0679	17.037	22.68	112.9	15.79	172.12
4	4.9663	17.653	22.196	106.04	16.691	166.21
5	5.1941	17.652	23.034	117.16	16.754	178.26
	5.08432	16.1296	22.3238	115.63	16.8678	174.492

Table 205 V2_03_TRC-Tracking Statistics

#Iterations	ORB Extraction	Pose Prediction	LM Track	New KF decision	Total Tracking
1	7.4388	1.7772	7.6982	0.082969	18.467
2	9.1328	2.2248	7.578	0.10101	20.843
3	9.33	2.178	6.7	0.097	20.58
4	9.16	2.2163	7.3292	0.091716	20.445
5	9.33	2.29	7.38	0.108	20.984
	8.87832	2.13726	7.33708	0.096139	20.2638

Discussion

This scheme is also successful in deleting keyframe and map point objects created by the system. However, this scheme does not improve performance as intended, rather it takes more time compared to the other two implementations. The following discussion compares the performance of this implementation with the last two and hypothesizes the reasons for the nature of the performance.

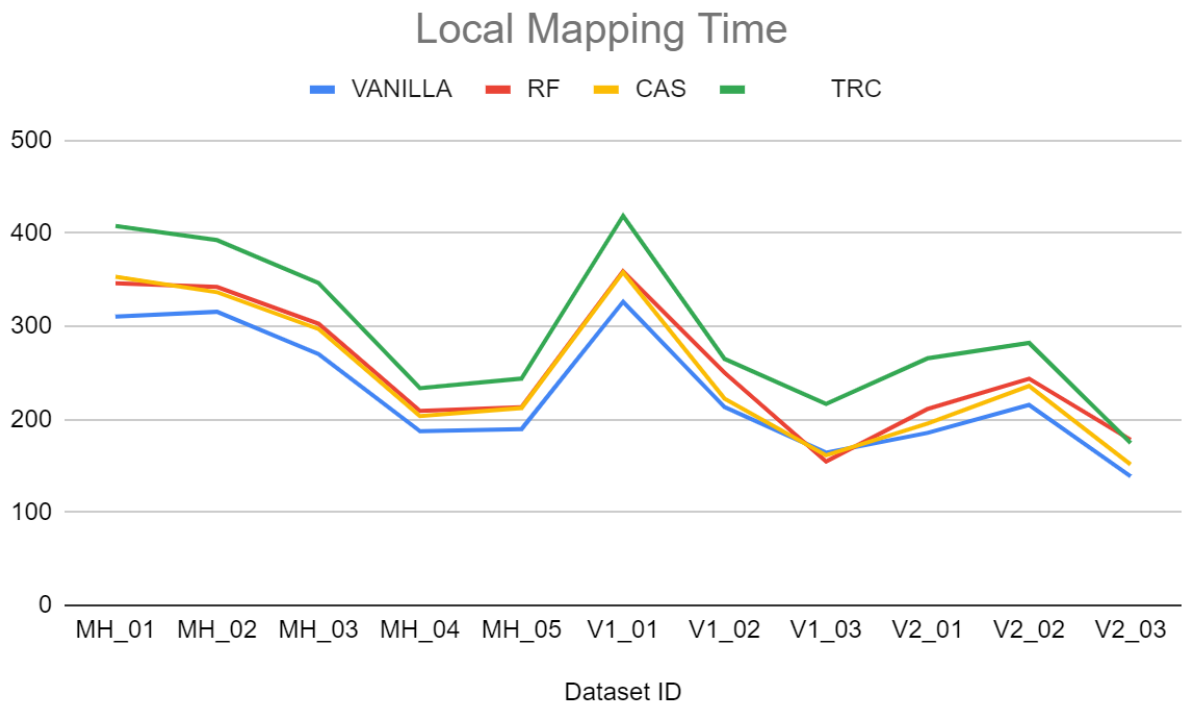
The graph below depicts the local mapping execution times plotted with respect to the 11 datasets. ORBSLAM3 without deletion demarcated by the blue line performs the best taking the least amount of time, followed by the reference counting implemented with CAS (yellow line), reference counting implemented with mutex locks (red line) and finally the reference counting implemented using thread-based counts (green line).

This is in line with expectations as reference counting will slow down the execution of functions due to the constant increments and decrements. Amongst the reference counted implementations, the CAS based reference counting is generally faster. This is not surprising as CAS operations will speed up counting for local mapping operations when compared to using mutexes.

The reference counting using thread-based counts is the slowest of all the implementations, this could be because of the look up time within the hash map associated with each keyframe and map point. This extra cost would add up over time with multiple increments and decrements.

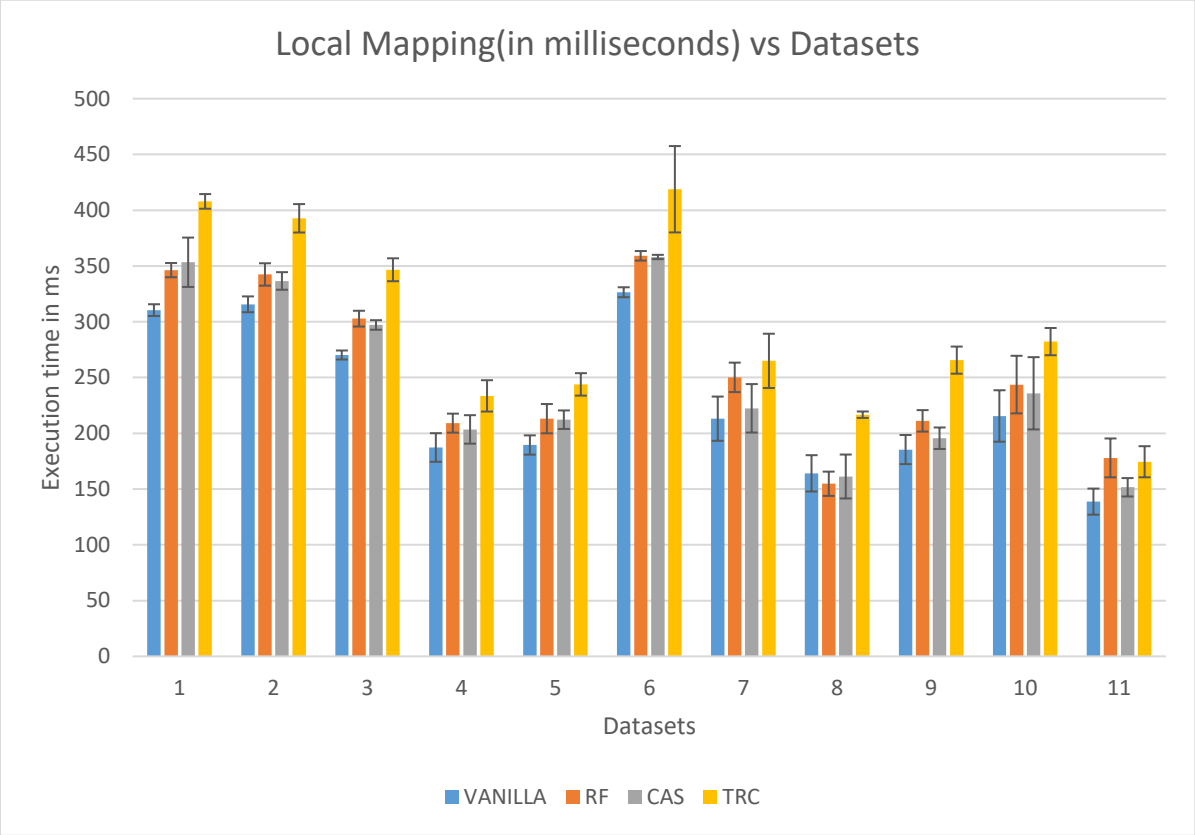
Possibly, the degree of concurrency exhibited by the system is not enough to gather the benefits offered by this form of reference counting. As a result, the system takes more time to reference count with this scheme.

Figure 26 Implementation specific Local Mapping Statistics for Thread Specific Reference Counting



The below figure represents the above with error bars.

Figure 27 Implementation specific Local Mapping Statistics for Thread Specific Reference Counting (with error bars)



The graph below depicts the tracking execution times plotted with respect to the 11 datasets. ORBSLAM3 without deletion demarcated by the blue line performs the best, taking the least amount of time, followed by the reference counting implemented with CAS (yellow line), reference counting implemented with mutex locks (red line) and finally reference counting implemented using thread-based counts (green line).

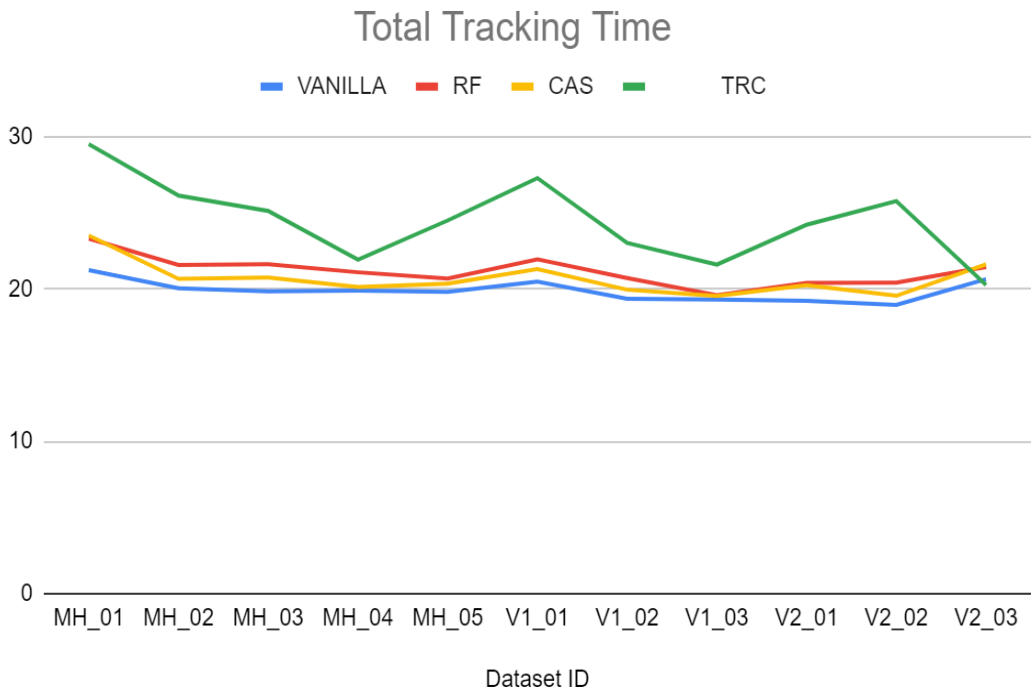
Just like local mapping, this is expected as reference counting will slow down operations due to constant increments and decrements. The CAS implementation is faster than the mutex locked

implementation. This is not surprising as CAS operations will speed up counting when compared to using mutexes.

The reference counting using thread-based counts takes the maximum amount of time in all the implementations. This could be because of the time taken to find a key and increment/ decrement the respective count within the hash map associated with each keyframe and map point as opposed to the increment/decrement of an integer.

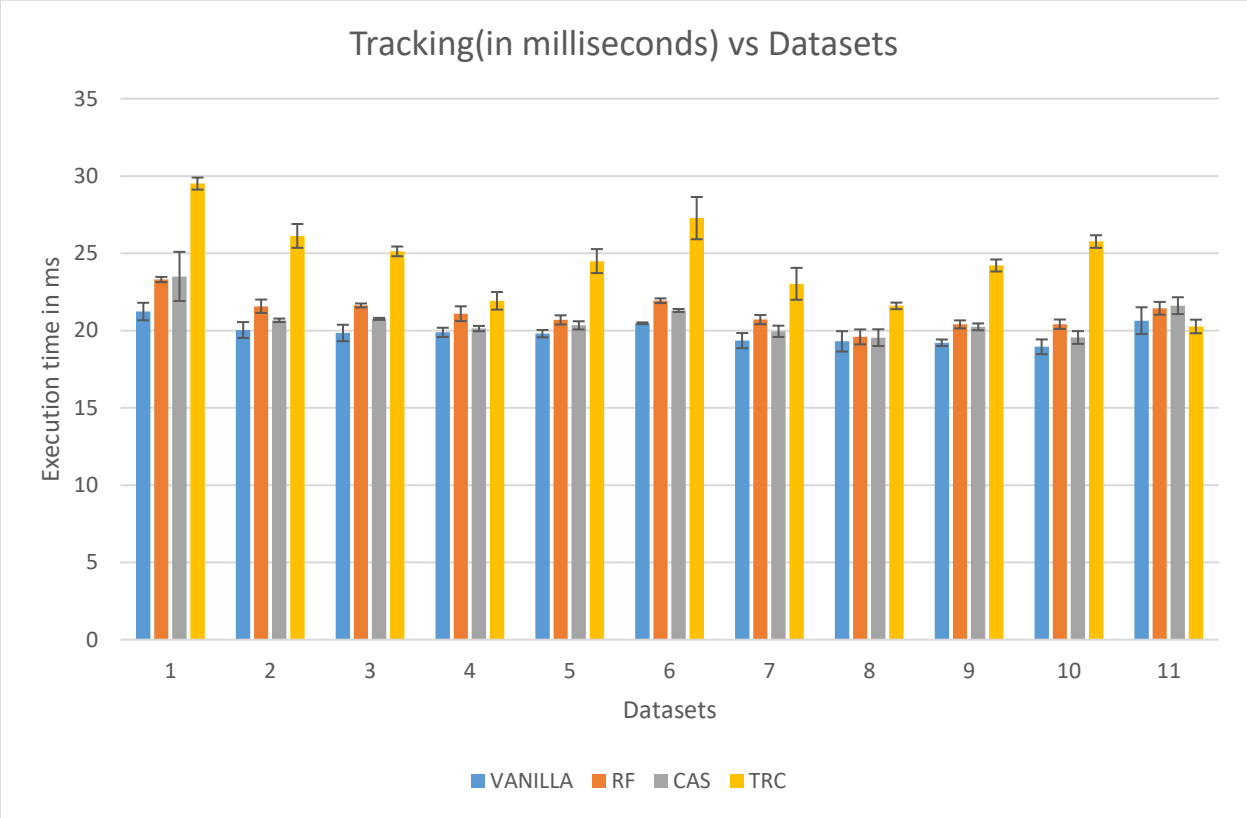
This added cost would increase over time with multiple executions of the reference counted functions. The time taken to perform the counts possibly outweighs the benefits provided by this scheme of reference counting.

Figure 28 Implementation specific Tracking Statistics for Thread Specific Reference Counting



The below figure represents the above with error bars.

Figure 29 Implementation specific Tracking Statistics for Thread Specific Reference Counting (with error bars)



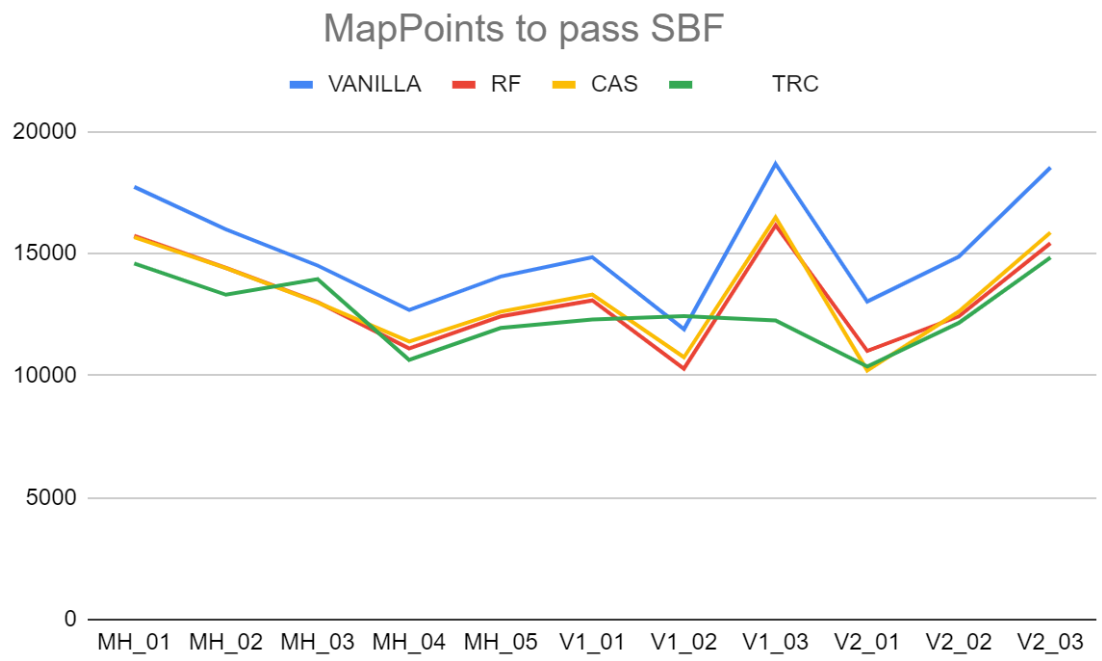
The graph below depicts the total number of map points marked bad plotted with respect to the 11 datasets. ORBSLAM3 without deletion is demarcated by the blue line marking the highest number of map points, reference counting implemented with CAS (yellow line), the reference counting implemented with mutexes (red line) and finally the reference counting implemented with thread-based counts (green line).

This behavior might be explained by the fact that the system spends a good amount of time to accommodate the reference counts. These computations along with the constraints of processing

data in real time might result in a lesser number of map points being marked bad. On some points, the thread-based reference counting implementation performs better than any of the other implementations including ORBSLAM3 without any deletion scheme, this must be investigated in the future.

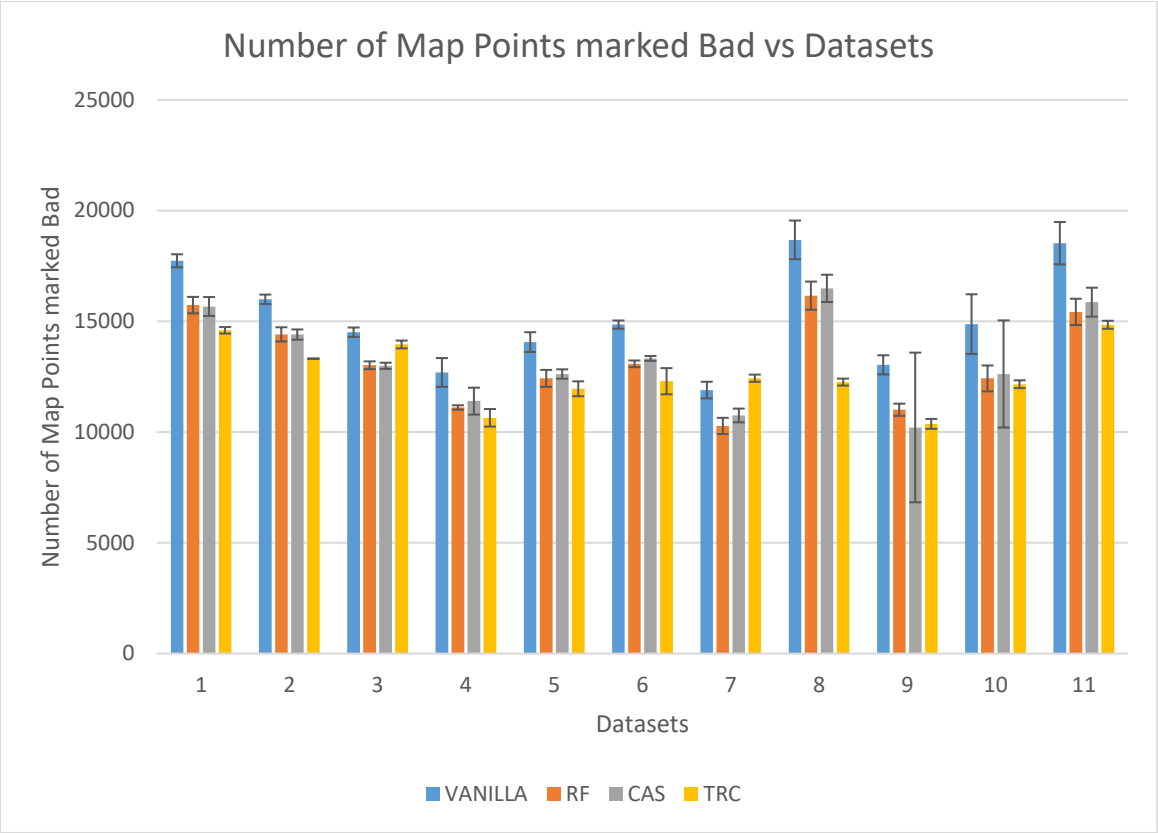
The general decrease in the number of marked map points for the thread-based scheme might be explained by the increase in time taken for reference counting. This is most likely due to the time taken to find the thread id key and update the corresponding count within the hash map belonging to a keyframe/map point. For dataset V1_02, the number of map points marked by the new scheme is more than the number of map points marked by any implementation. The cause for this must be investigated in the future.

Figure 30 Map points marked bad for Thread Specific Reference Counting



The figure below represents the above with error bars.

Figure 31 Map points marked bad for Thread Specific Reference Counting (with error bars)

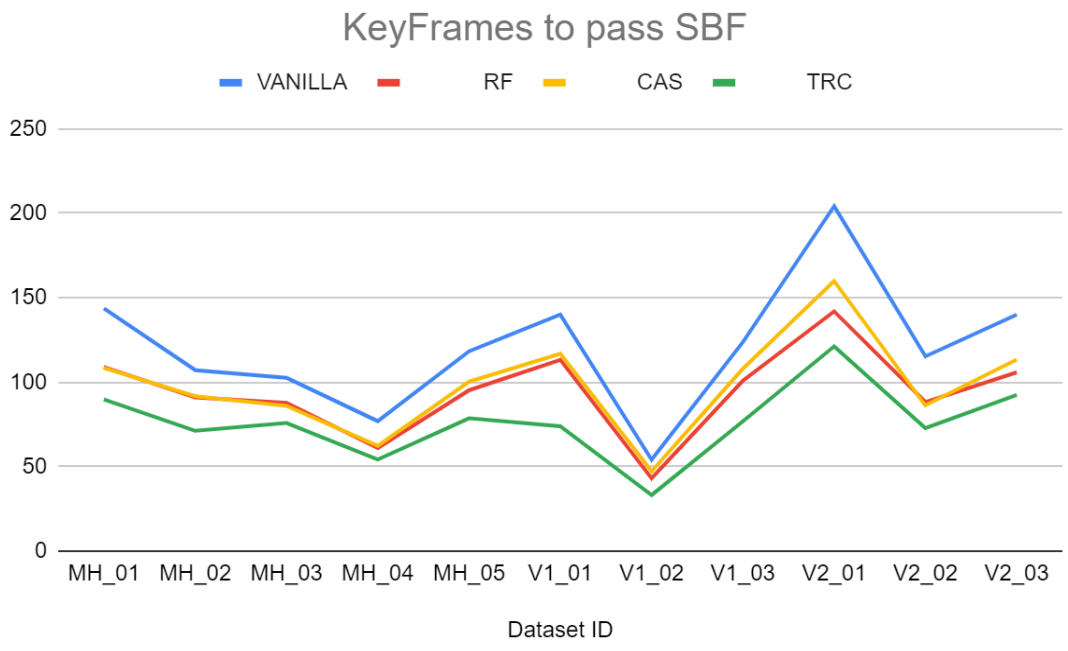


The graph below depicts the number of keyframes marked bad plotted with respect to the 11 datasets. ORBSLAM3 without deletion demarcated by the blue line performs the best marking the highest number, followed by the reference counting implemented with CAS (yellow line) reference counting implemented with mutex locks (red line) and finally reference counting implemented using thread-based counts (green line). This is not surprising as CAS operations will speed up counting when compared to using mutexes.

This slowdown in the behavior of the reference counted implementations might be explained by the fact that the system spends a good amount of time to accommodate the reference counts. These computations along with the constraints of processing data in real time might result in a lesser number of keyframes being marked bad.

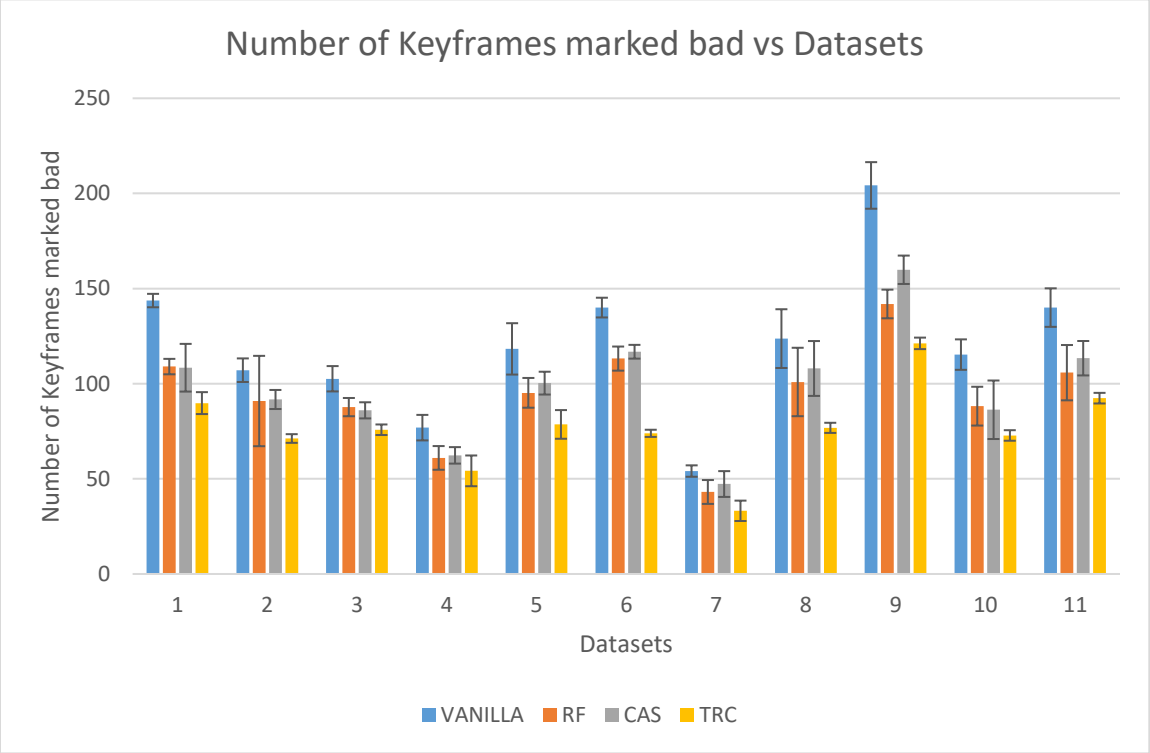
The implementation based on per thread reference counts is slower than any of the other implementations. This could be because of the extra time associated with finding a key and updating its corresponding count in a hash map as opposed to incrementing/decrementing an integer.

Figure 32 Number of Keyframes marked bad.



The figure below represents the above with error bars.

Figure 33 Number of Keyframes marked bad (with error bars)



The graph below depicts the number of keyframes deleted, plotted with respect to the 11 datasets. Reference counting implemented with CAS (red line) performs the best, followed by reference counting implemented with mutex locks (blue line) and finally reference counting implemented with mutex thread-based counts (yellow line).

This behavior might be because of the load created by the reference counting infrastructure along with the real time constraints of the system affects the addition of keyframes to the map. The decrease in the number of keyframes deleted by the implementation making use of the thread-based count could be attributed to the extra time taken for updating the count within the hash map

belonging a keyframe or map point. This might outweigh the degree of concurrency present in the system, thereby resulting in a net increase of time for reference counting.

Figure 34 Keyframes deleted for Thread Specific Reference Counting

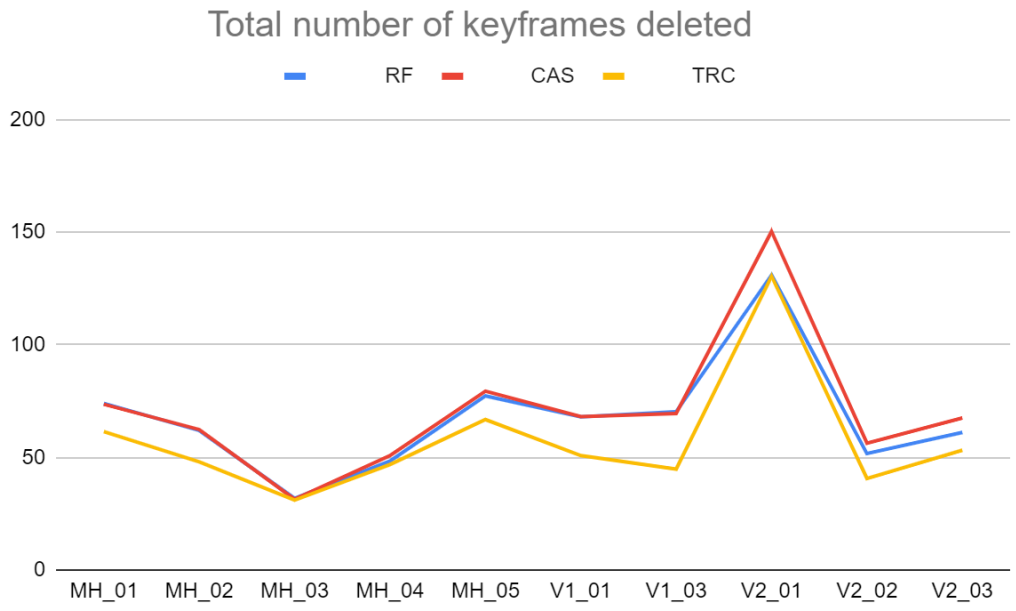
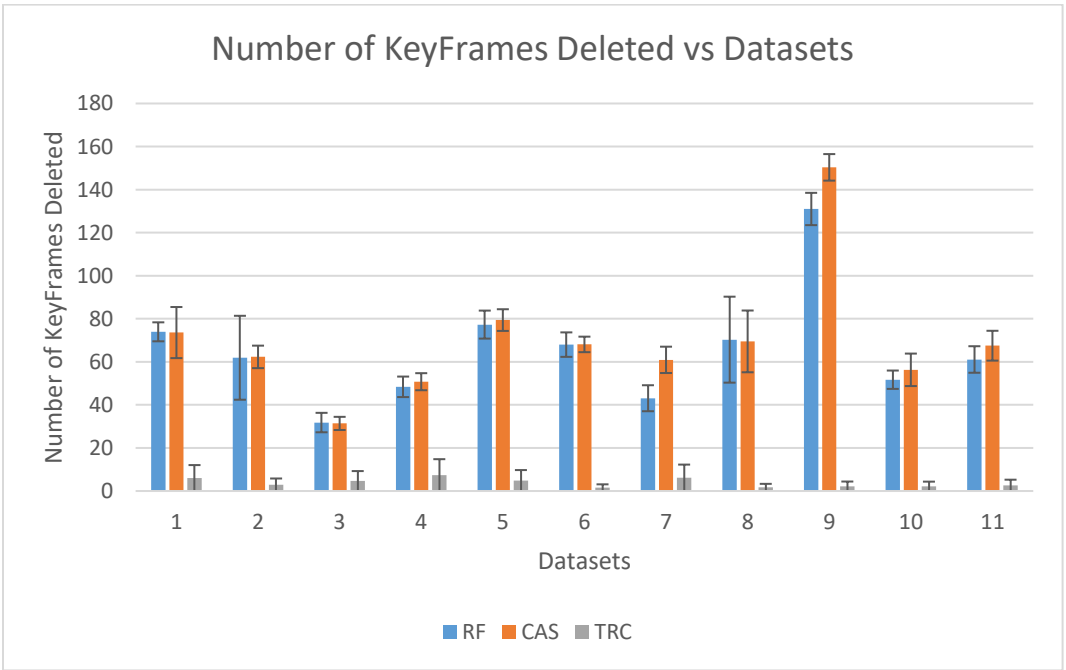


Figure 35 Keyframes deleted for Thread Specific Reference Counting (with error bars)

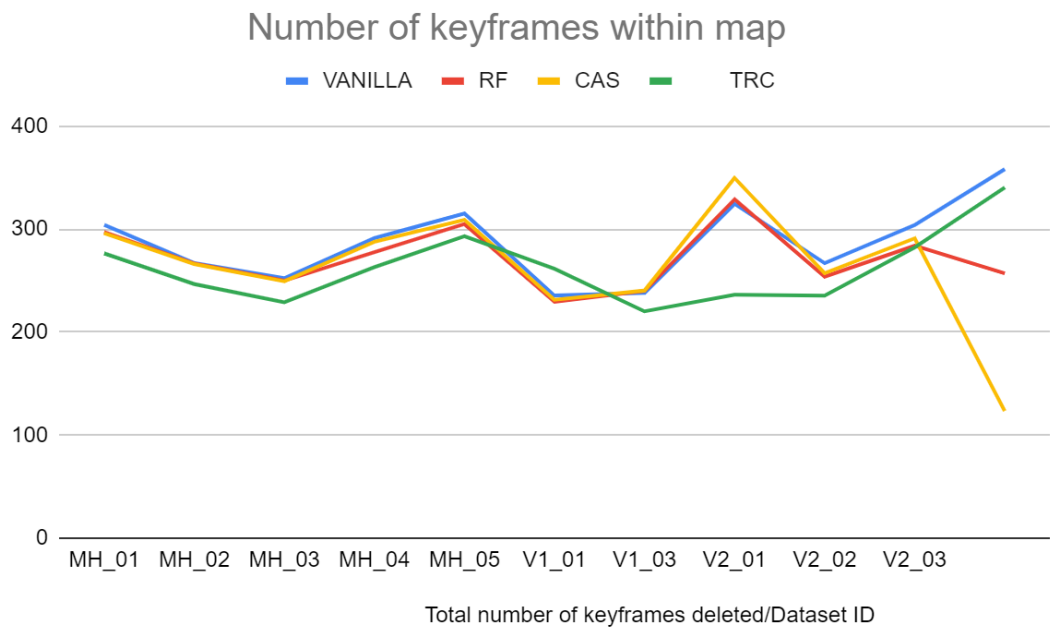


The graph below depicts the number of keyframes within the map plotted with respect to the 11 datasets, ORBSLAM3 without deletion demarcated by the blue line performs the best, capturing many keyframes within the map, reference counting implemented with mutex locks (red line) and finally reference counting implemented with mutex thread-based counts (green line).

This behavior might be because of the load created by the reference counting infrastructure along with the real time constraints of the system affects the addition of keyframes to the map. However, the yellow line has several spikes above plain ORBSLAM3 and dips below the mutex implemented count. This must be investigated in the future.

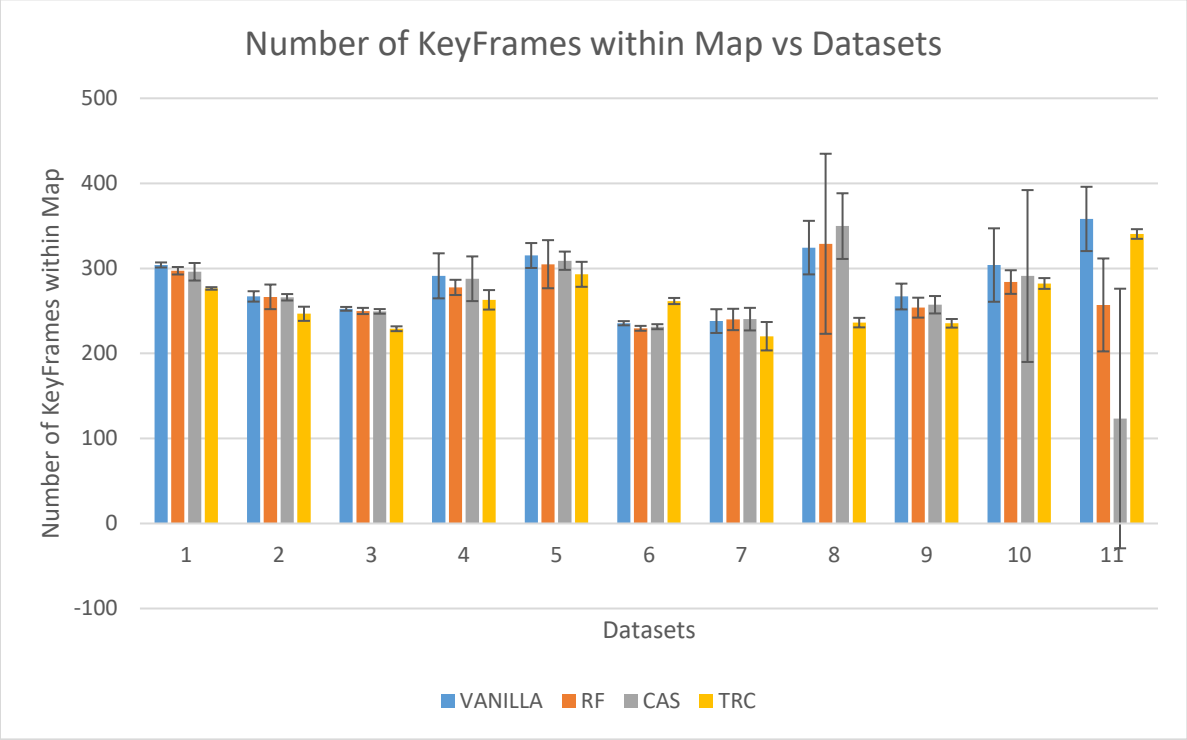
The implementation that makes use of the thread-based count is the least performant. This might be due to the extra time taken to find a key and update its respective count within the hash map belonging to a keyframe/map point.

Figure 36 Keyframes within Map for Thread Specific Reference Counting



The figure below represents the above with error bars.

Figure 37 Keyframes within Map for Thread Specific Reference Counting (with error bars)



The graph below depicts the number of map points within the map plotted with respect to the 11 datasets. ORBSLAM3 without deletion demarcated by the blue line performs the best, capturing many map points within the map, reference counting implemented with mutex locks (red line) and finally reference counting implemented with mutex thread-based counts (green line).

This behavior might be because of the load created by the reference counting infrastructure along with the real time constraints of the system affects the addition of map points to the map. Also, the yellow line that denotes CAS has several spikes above plain ORBSLAM3 and dips below the mutex implemented count. The cause of this behavior must be investigated in the future.

The new scheme performs poorly compared to the other schemes, this could be explained by the extra time taken to find a key and update the corresponding count outweighing the degree of concurrency of the system. However, the number of map points within the map is higher than any other implementation for the dataset named V1_02. The cause for this must be investigated in the future.

Figure 38 Map points within Map for Thread Specific Reference Counting

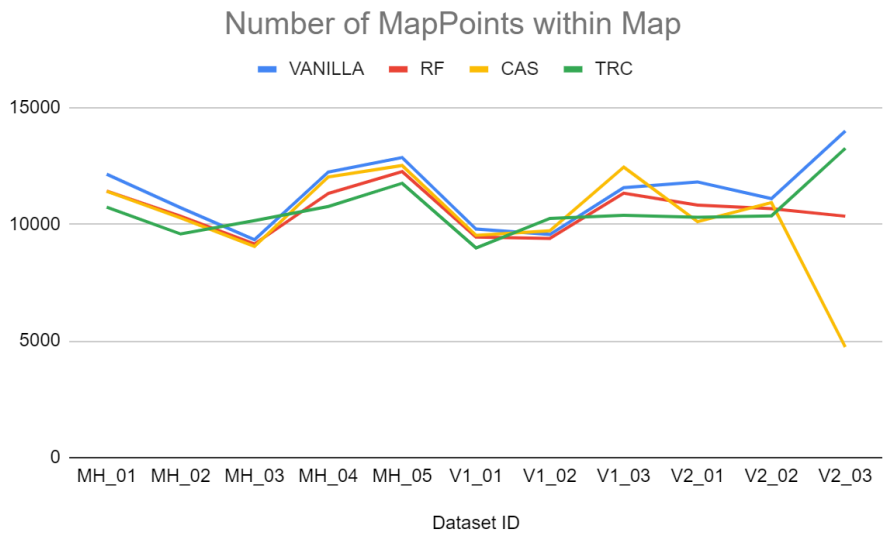
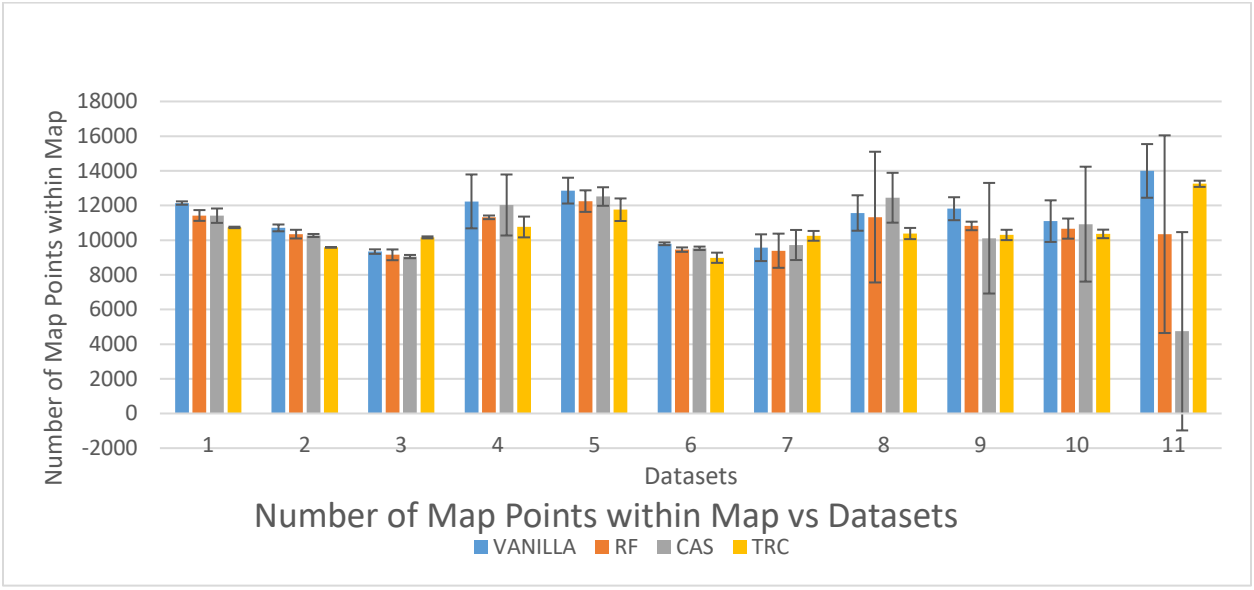


Figure 39 Map points within Map for Thread Specific Reference Counting (with error bars)

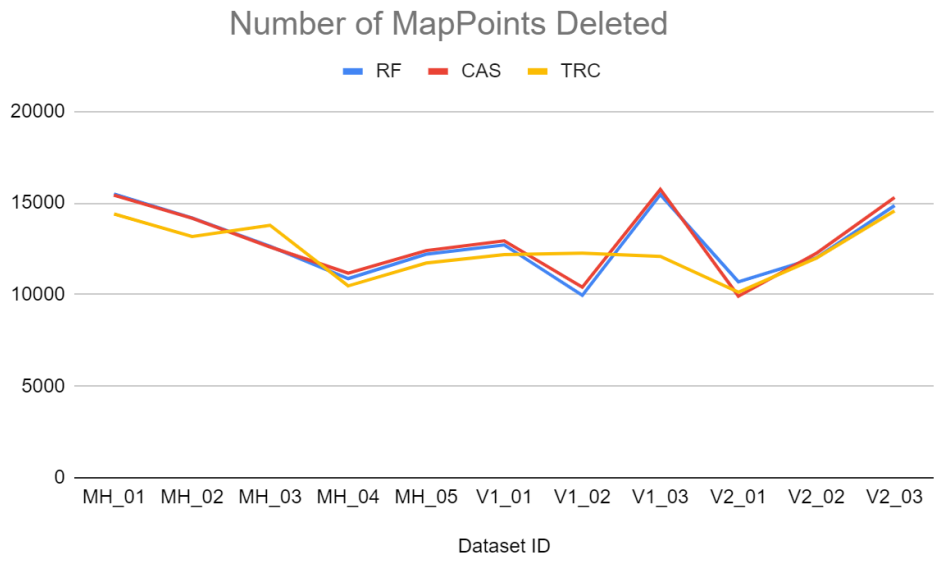


The graph below depicts the number of map points deleted plotted with respect to the 11 datasets. The CAS scheme (red line) performs the best, deleting the greatest number of map points, followed by reference counting implemented with mutex locks (blue line) and finally reference counting implemented with mutex thread-based counts (yellow line).

The thread-based count implementation is the least performant implementation. This could be explained by the extra time taken to find a key and update the corresponding count outweighing the degree of concurrency of the system. However, the number of deletions is higher than any other implementation for the dataset named V1_02. The cause for this must be investigated in the future.

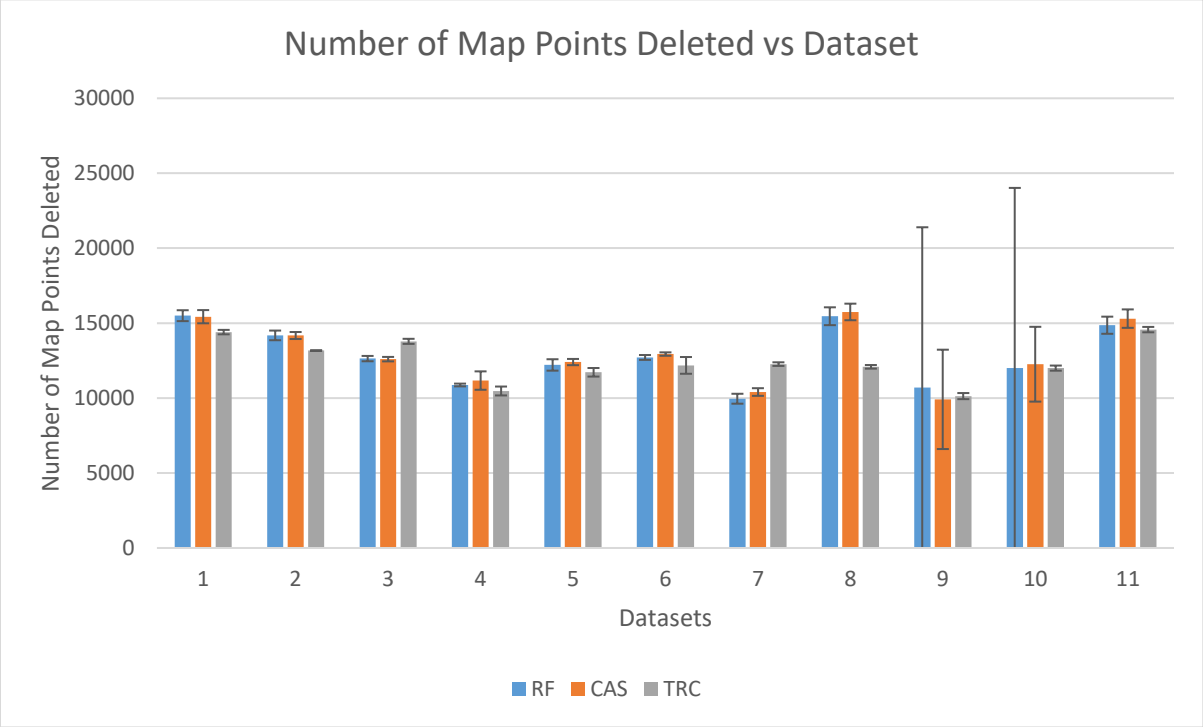
To sum it up, thread-based reference counting is not as performant as any of the other implementations for the reasons mentioned above.

Figure 40 Map points deleted for Thread Specific Reference Counting



The figure below represents the above with error bars.

Figure 41 Map points deleted for Thread Specific Reference Counting (with error bars)



Conclusion

The ORBSLAM3 codebase was instrumented with reference counting to facilitate safe deletion of heap allocated memory. Thread safe reference counting was carried out with the help of three different schemes. The first scheme made use of mutexes, the second scheme used compare and swap to facilitate reference counting and the third scheme involved the usage of thread specific reference counts.

Even though all the schemes are successful in deleting heap allocated objects, the compare and swap implementation is generally more performant than the other two implementations in terms of time, input data processed as well as the number of heap allocations freed. It is followed by reference counting with mutexes and finally the reference counting with separate counts for each thread. The CAS implementation outperforms the mutex implementation due to the threads not being put to sleep/blocked.

One of the possible reasons for the low performance exhibited by the scheme that makes use of separate counts for each thread, is the look up time and value modification time associated with the hash maps belonging to reference counted objects.

This could be improved by coming up with a more efficient mapping scheme between the thread identification objects and the reference counts. The mapping scheme may be improved by modifying the thread interface provided by the C++ standard library to incorporate features friendly to reference counting on a per thread basis.

Another solution would be to use and adapt the pthread interface. However, this would involve swapping all the functionality offered by the std thread interface with that of the pthread interface.

Deletion of heap allocated objects in ORBSLAM3 is not easy due to the cyclic nature of references and the number of threads manipulating the references to keyframes and objects. The schemes capable of deletion may have to be reinforced by testing with several more datasets to be highly robust. This would be necessary as the nature of the dataset would determine changes in the call graph for the program. Another method to make the reference count more robust would be testing in conditions where tracking, local mapping and loop closing work faster than usual.

References

[1] D. M. Georg Klein, "Parallel tracking and mapping for small AR workspaces," in *6th IEEE and ACM international symposium on mixed and augmented reality*, 2007.

[2] R. C. B. Martin A Fischler, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," in *Communications of the ACM*, 1981.

[3] T. D. Edward Rosten, "Machine Learning for High-Speed Corner Detection," in *Computer Vision – ECCV*, 2006.

[4] R. I. H. a. A. Zisserman, ".Multiple View Geometry in Computer Vision," *Cambridge University Press*, 2004.

[5] R. C. S. a. P. Cheeseman, "On the representation and estimation of spatial uncertainty," in *International Journal of Robotics Research*, 1986.

[6] S. J. L. A. J. D. Richard A Newcombe, "DTAM: Dense tracking and mapping in real-time," in *International Conference on Computer Vision*, 2011.

[7] I. M. Simon Baker, "Lucas-Kanade 20 Years On: A Unifying Framework," in *International Journal of Computer Vision* , 2004.

[8] J. E. a. T. S. a. D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," in *European Conference on Computer Vision*, 2014.

[9] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige and W. Burgard, "g2o: A General Framework for Graph Optimization," [Online]. Available: https://openslam-org.github.io/g2o.html.

[10] N. E. F. E. W. B. D. C. Jürgen Sturm, "A benchmark for the evaluation of RGB-D SLAM systems," *International Conference on Intelligent Robots and Systems*, 2012.

[11] V. R. K. K. G. B. Ethan Rublee, "ORB: An efficient alternative to SIFT or SURF," in *International Conference on Computer Vision*, 2011.

[12] J. M. M. M. J. D. T. Raúl Mur-Artal, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," in *IEEE Transactions on Robotics*, 2015.

[13] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," in *International Journal of Computer Vision*, 2004.

[14] T. T. L. V. G. Herbert Bay, "SURF: Speeded Up Robust Features," in *Computer Vision – ECCV* , 2006.

[15] V. L. C. S. a. P. F. Michael Calonder, "BRIEF: Binary Robust Independent Elementary Features," in *Computer Vision – ECCV*, 2010.

[16] D. Galvez-Lopez, "DBoW2 Library," [Online]. Available: https://webdiis.unizar.es/~dorian/doc/dbow2/index.html.

[17] F. M.-N. a. P. F. V. Lepetit, "EPnP: An accurate O(n) solution to the PnP problem," *International Journal of Computer Vision*, vol. 81, no. 2, pp. 155-166, 2009.

[18] N. D. D. S. J. S. D. C. Vladyslav Usenko, "Visual-Inertial Mapping with Non-Linear Factor," in *IEEE ROBOTICS AND AUTOMATION LETTERS*, 2019.

[19] W. B. a. G. D. T.] M. Mazuran, "Nonlinear factor recoveryfor long-term SLAM," *The International Journal of Robotics Research*, vol. 35, pp. 50-72, 2015.

[20] "Kullback-Leibler Divergence," [Online]. Available: https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence.

[21] J. N. G. W. A. S. Michael Burri, "The EuRoC micro aerial vehicle datasets," *The International Journal of Robotics Research,Thomas Schneider,Joern Rehder,Sammy Omari,* vol. 35, no. 10, pp. 1157-1163, 2016.

[22] J. D. T. Raul Mur-Artal, "ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras," in *IEEE Transactions on Robotics*, 2016.

[23] A. G. a. P. L. a. R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," *Conference on Computer Vision and Pattern Recognition (CVPR),* 2012.

[24] J. D. T. Raul Mur-Artal, "Visual-Inertial Monocular SLAM with Map Reuse," in *IEEE Robotics and Automation Letters*, 2017.

[25] S. L. M. B. R. S. P. F. Stefan Leutenegger, "Keyframe-Based Visual-Inertial SLAM Using Nonlinear Optimization," in *The International Journal of Robotics Research*, 2014.

[26] M. S. Chris Harris, "A combined corner and edge detector," *Alvey Vision Conference,* 1988.

[27] S. O. M. H. R. S. Michael Bloesch, "Robust Visual Inertial Odometry Using a Direct EKF-Based Approach," in *International Conference on Intelligent Robots and Systems*, 2015.

[28] "https://en.wikipedia.org/wiki/Extended_Kalman_filter," [Online].

[29] C. T. J. Shi, "Good features to track," *International Conference on Pattern Recognition,* pp. 593-600, 1994.

[30] T. K. B. D. Lucas, "An iterative image registration technique with an application to stereo vision," *Proceedings of International Joint Conferences on Artificial Intelligence,* pp. 24-28, 1981.

[31] R. E. J. J. G. R. J. M. M. M. Carlos Campos, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM," in *IEEE Transactions on Robotics*, 2021.

[32] T. G. D. V. U. J. S. C. David Schubert, "The TUM VI Benchmark for Evaluating Visual-Inertial Odometry," *International Conference on Intelligent Robots and Systems*, 2018.

[33] M. A. Y. C. L. C. Antoni Rosinol, "Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping," in *IEEE International Conference on Robotics and Automation*, 2020.

[34] T. S. M. P. a. L. C. A. Rosinol, "Incremental Visual-Inertial 3D Mesh Generation with Structural Regularities," *Intl. Conf. on Robotics and Automation*, 2019.

[35] Z. T. M. F. R. S. a. J. N. H. Oleynikova, "Incremental 3d euclidean signed distance fields for onboard mav planning," *Intl. Conf. on Intelligent Robots*, pp. 1366-1373, 2018.

[36] S. Leutenegger, "OKVIS2: Realtime Scalable Visual-Inertial SLAM with Loop Closure," 2022.

[37] M. C. Y. S. Stefan Leutenegger, "BRISK: Binary Robust invariant scalable keypoints," *International Conference on Computer Vision*, 2011.

[38] "Hamming Distance," [Online]. Available: https://en.wikipedia.org/wiki/Hamming_distance#:~:text=9%20Further%20reading-,Definition,the%20corresponding%20symbols%20are%20different..

[39] M. Zhong, "A Survey of Garbage Collection Techniques," 5 May 1998. [Online]. Available: https://pages.cs.wisc.edu/~zhong/termproj_surveyGC.html.

[40] M. W. Hans-Juergen Boehm, "Garbage collection in an uncooperative environment," *Software: Practice and Experience,* vol. 18, no. 9, pp. 807-820, 1988.

[41] "https://en.cppreference.com/w/cpp/memory/shared_ptr," [Online].

[42] "opencv," [Online]. Available: https://github.com/opencv/opencv.

[43] "Eigen," [Online]. Available: https://eigen.tuxfamily.org/index.php?title=Main_Page.

[44] "std::atomic_compare_exchange_strong," [Online]. Available: https://cplusplus.com/reference/atomic/atomic_compare_exchange_strong/.

[45] P. L. S. S. Tong Qin, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," in *IEEE Transactions on Robotics*, 2018.

[46] "voxblox," [Online]. Available: https://github.com/ethz-asl/voxblox.