# Self Localization using Visual-Inertial SLAM for navigation of human workers

Saqib Azim

May 15, 2020

**Abstract**

This research presents a self-localization method aimed at localizing and navigating a human worker in a prebuilt map of a target environment using the technology of visual-inertial SLAM (Simultaneous Localization and Mapping) framework. We aim to create a tool in the form of an easy-to-use smartphone application which can be used by a human worker for navigation in a factory-based environment. Most of the existing traditional SLAM methods are based on the assumption of static-based environments. We include deep learning based object detection and segmentation module to improve the performance of visual SLAM in real-world environment (with moving people, objects, etc). Using dynamic segmentation module in the visual SLAM pipeline mitigates the effects of dynamic moving objects in the environment and improves the tracking accuracy as well as makes the environment map more stable and re-usable for long-term applications. Through our experiments, we provide comparison between the performance of our approach and traditional SLAM framework on public datasets and validate that using dynamic object segmentation improves visual SLAM accuracy and robustness in localization performance in real-world environments.

# Contents

**1 Introduction** **3**
1.1 Motivation and Objective . . . 3
1.2 Progress in research . . . 3
1.3 Research Targets . . . 3

**2 Related Work** **4**
2.1 Visual SLAM . . . 4
2.2 Object Detection and Segmentation Framework . . . 4

**3 Research Method** **5**
3.1 Feature Extraction Module . . . 6
3.2 Potential Dynamic Content Segmentation using CNN-based network . . . 6
3.3 Camera Pose Estimation . . . 7
3.4 Mapping Module . . . 8
3.5 Loop Detection and Closure . . . 9
3.6 Re-Localization and Navigation . . . 10

**4 Evaluation Metrics** **11**

**5 Datasets** **12**
5.1 TUM RGB-D Dataset . . . 12
5.2 KITTI Dataset . . . 12

**6 Experiments and Results** **13**
6.1 Dynamic object segmentation module - Mask RCNN . . . 13
6.1.1 TUM RGB-D dataset . . . 13
6.1.2 KITTI dataset . . . 15
6.1.3 Experimental Conclusion . . . 16

**7 Problems encountered and bottlenecks** **16**

**8 Safety Considerations** **17**

**9 Environmental issues considered** **17**

**10 Benchmark of this research** **17**

**11 Consideration of IPR status** **17**
11.1 Intellectual property owned by other organisations (domestic/foreign patent) . . . 17

**12 Conclusion** **17**
12.1 Summary . . . 17
12.2 Future issues . . . 17

# 1 Introduction

## 1.1 Motivation and Objective

In the manufacturing sector, the retirement of the expert workers and labourers who have been supporting the strong industry for so many years is a cause of grave concern for the future. We strongly need technology in the form of robotics and automation to support human workers. Hitachi has been at the forefront in tackling this social issue and has been developing various systems for supporting human workers. One of these areas which requires some attention is easy-to-use digitized localization and navigational facilities for novice workers in factories and warehouses. Expert workers are being retired and this leads to requirement of new non-expert workers who are not very much familiar with working environments and locations. In the future of industries and factories, most tasks are expected to be carried out by robots instead of human workers. These human workers might be required for various tasks at multiple locations. It might be difficult for a new human worker to become familiar with the details of multiple different locations in terms of checking and maintenance.

In order to address this challenge, we are trying to create a localization and navigational application using the technology of visual-inertial SLAM. Human workers generally have smartphone or cameras which can provide visual image sequences. This image sequence can then be processed using a smartphone application. The key idea is to have a map for all the target locations (created using SLAM technology as a part of initial setup). When a human worker wants to localize himself or needs a navigational help in a factory, they can load the map specific to the factory or location. The localization algorithm can then estimate and provide the location of the human worker using the visual data and the inertial-measurement-unit (IMU) data from the smartphone.

The traditional SLAM approaches are based on the assumption of static environment. Thus, in dynamic environment (dynamic moving objects), these approaches will perform less accurately. Due to the presence of dynamic objects, the localization of the camera becomes less accurate. In addition to this, the environment map created with the static approach will include the erroneous map points from the dynamic objects which, in turn, increases the error in the system during localization and navigation. Therefore, by including dynamic segmentation module, this research helps to improve the localization accuracy in real-world environments.

## 1.2 Progress in research

We verified the feasibility of the SLAM technology for human localization and navigation in the factory, and we got some prospect of realization of this technology in Kenkyu Kaihatu shien: R&D support fund (last year) from HISYS. We are looking for business needs of this technology and customers through demonstrations and conference presentations.

## 1.3 Research Targets

The research targets for this project can be summed up in following points:

1. To develop a technology for self-location estimation using video for human workers in factory environment
2. To improve the localization and mapping performance of SLAM in dynamic real-world environments
3. To create a smartphone application for demonstrating the localization and navigational use of this technology targeted as a helping tool for human workers in factories

# 2 Related Work

## 2.1 Visual SLAM

Visual SLAM has drawn the attention of researchers over the past decade because of its application in robotics and automation. Monocular SLAM was initially solved by filtering [1], [2] every frame in order to jointly estimate the map feature locations and the camera pose. On the other hand, currently used keyframe-based approaches estimate the map using only certain important and non-redundant frames (keyframes) which allows the system to perform more costly but accurate bundle adjustment optimizations. Strasdat et al. [3] demonstrated that keyframe-based techniques are more accurate than filtering for nearly the same computational cost. The PTAM [4] introduced for the first time two parallel threads for visual SLAM framework: tracking and mapping. This became the benchmark as well as fundamental baseline for various future SLAM-based techniques.

Visual SLAM can be classified into feature-based and direct methods. Feature-based methods rely on salient keypoint matching and can only estimate a sparse or semi-dense reconstruction of the environment, whereas direct methods estimate, in principle, a fully or semi-dense map by minimization of the photometric error directly over image pixels [5], by exploiting the brightness information of all the image pixels. In the category of direct SLAM based methods, LSD-SLAM [5] can build large-scale semi-dense maps but they do not perform loop detection and the camera localization accuracy is significantly lower compared to current state-of-the-art feature-based methods. The direct based method saves computing resources in tracking and feature matching, however, the insensitivity to features is a fatal weakness for its use in practical SLAM applications. Direct methods are, in general, more sensitive to dynamic objects in the scene.

Feature points are essential in order to build a complete and robust SLAM framework. Feature extraction and matching ensures the accuracy of camera pose estimation in the SLAM tracking. Feature-based method can extract more effective information from visual images, such as semantics, object recognition, feature localization, etc. ORB-SLAM [6] proposed by Mur-Artal et al., is one of the fundamental and successful SLAM techniques based on the feature tracking method. It can enable the system to run for a long time under large scenes and large loops, thus ensuring the global consistency of the trajectory and the 3D map. Though one of the shortcomings of ORB-SLAM is in dealing with dynamic environments.

Dynamic objects are classified as spurious data in SLAM and therefore should not be included in the map as well as should not be used for camera pose estimation. SLAM in dynamic environments has always been challenging because traditional SLAM theory and vast majority of existing approaches assume a static environment. Its challenges mainly come from two aspects: first, it is difficult to define a dynamic object from planar pixel and secondly dynamic objects are not easily detected and tracked. For dealing with dynamic environments, there have been a few exciting approaches in recent years. Chen et al. [7] and CD-SLAM [8] proposed by Picker et al are few of the earlier approaches trying to deal with this issue. Dynamic SLAM [9] uses SSD object detection framework [10] for detecting potential dynamic objects in an additional thread. The object detection thread produces bounding-boxes around detected dynamic objects. This information is further forwarded to ORB-SLAM framework for tracking and mapping. One possible limitation of this approach is that bounding boxes around a dynamic object may also include additional portion of the scene which might contain useful feature points. Although one major advantage is that current state-of-the-art detectors such as YOLO [11] or SSD [10] can perform in real-time with acceptable accuracy.

## 2.2 Object Detection and Segmentation Framework

Recent advances in object detection and semantic segmentation methods based on CNN have led to many breakthroughs in numerous real-life problems. The object detection networks headed by R-CNN [12], such as SPP-Net [13], Fast R-CNN [14], Faster R-CNN [15] etc., use CNN to automatically learn features and avoid the limitation of designing manual features. Although these methods have high precision, it is very time-consuming because candidate frame extraction and object recognition are performed in two separate steps. Lots of approaches based on fully convolutional network (FCN) [16] have achieved state-of-the-art performance on different benchmarks of the semantic segmentation task. In this domain, Mask R-CNN [17] based on Faster R-CNN with an additional overhead network specially designed for segmentation, provides great results in terms of accuracy but its speed limitation is an issue. He et al. [17] report that MaskRCNN runs at 195 ms per image (~ 6 fps) on a NVIDIA Tesla M40 GPU.

## 3 Research Method

With the help of this research, we aim at solving two main challenges: firstly, we want to provide navigational help to human workers in factories using a smartphone application. Secondly, we aim towards improving traditional visual SLAM system which are mainly based on the assumption of static environment scenarios. Tackling the second part, we work with an existing visual SLAM module - OpenVSLAM [18] as the baseline and improve its localization performance by specifically dealing separately with dynamic (or moving) objects in the scene. Our system consists of a CNN-based dynamic object segmentation module which performs pixelwise image segmentation on each and every incoming frame and outputs a binary mask image detecting the pixels belonging to any of the predefined dynamic object categories (such as people, vehicles, etc.) in the frame as shown in Fig. 1. Each incoming video frame is also passed through a feature extraction module which extracts a set of keypoint features. These set of keypoints and detected dynamic object binary masks are forwarded to the visual SLAM module [18] which estimates the camera pose (position and orientation of the camera at the time of capturing of current frame) and simultaneously constructs the 3D map of the environment visible to the camera. The core idea behind performing image segmentation is to identify and mitigate the effect of dynamic moving objects in the scene, if any. In our case, we accomplish this by rejecting all the keypoints that belong to detected dynamic objects in the scene so that these keypoints are neither used in estimating the camera pose nor for creating the 3d map of the environment. Thus, this approach tends to use only those keypoints that potentially belongs to static objects in the scene. Contrary to our approach, most traditional visual SLAM approaches (which are based on static scene assumption) [4], [6], [5] only extract keypoints from each incoming frame followed by estimating the camera pose and 3D environment map, without explicitly tackling the issue of dynamic objects in the scene.

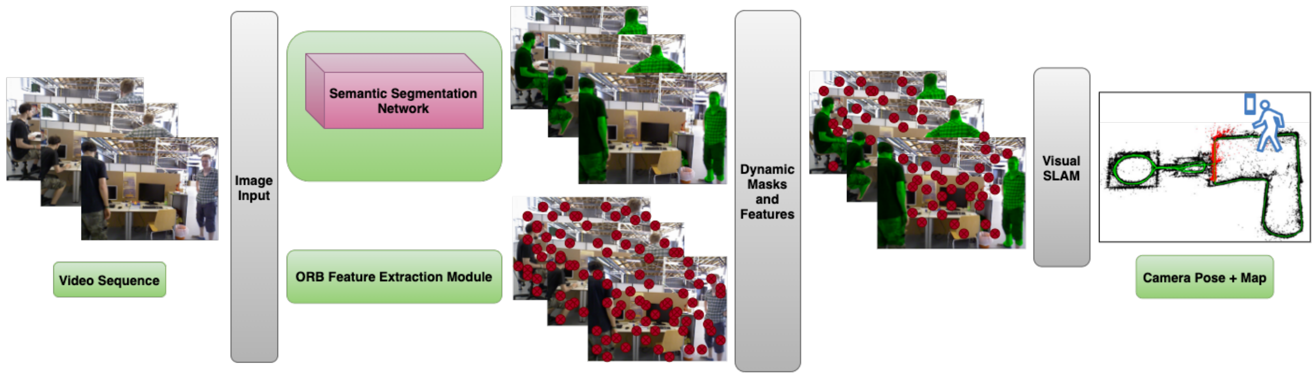

Video Sequence
Image Input
Semantic Segmentation Network
ORB Feature Extraction Module
Dynamic Masks and Features
Visual SLAM
Camera Pose + Map

**Figure 1:** System Overview of our approach: Incoming frames from monocular camera serve as the input data to the semantic segmentation module and the visual SLAM module. The output of semantic segmentation module is passed to the SLAM module which estimates the camera pose of every frame as well as constructs the 3D map of the environment

Our system mainly involves three threads that simultaneously run in parallel: tracking, mapping and global optimization thread as shown in Fig. 2. All these threads run in an asynchronous manner. The tracking module is responsible for localizing the camera by estimating its pose (position and orientation) with every incoming frame. It also takes the decision on whether a frame should be qualified as a keyframe (important and relevant frames) or not (based on certain conditions). Any frame marked as keyframe by the tracking module is passed on to the mapping and global optimization modules as they only process keyframes rather than all the frames. The mapping module involves estimation of 3D landmark positions with respect to world coordinate system (WCS) using only the keyframes. The global optimization thread performs multiple tasks sequentially: Firstly, it jointly optimizes the estimated camera poses and 3D map points using bundle adjustment (BA) technique [19]. Secondly, this thread checks for any potential loops (when camera returns

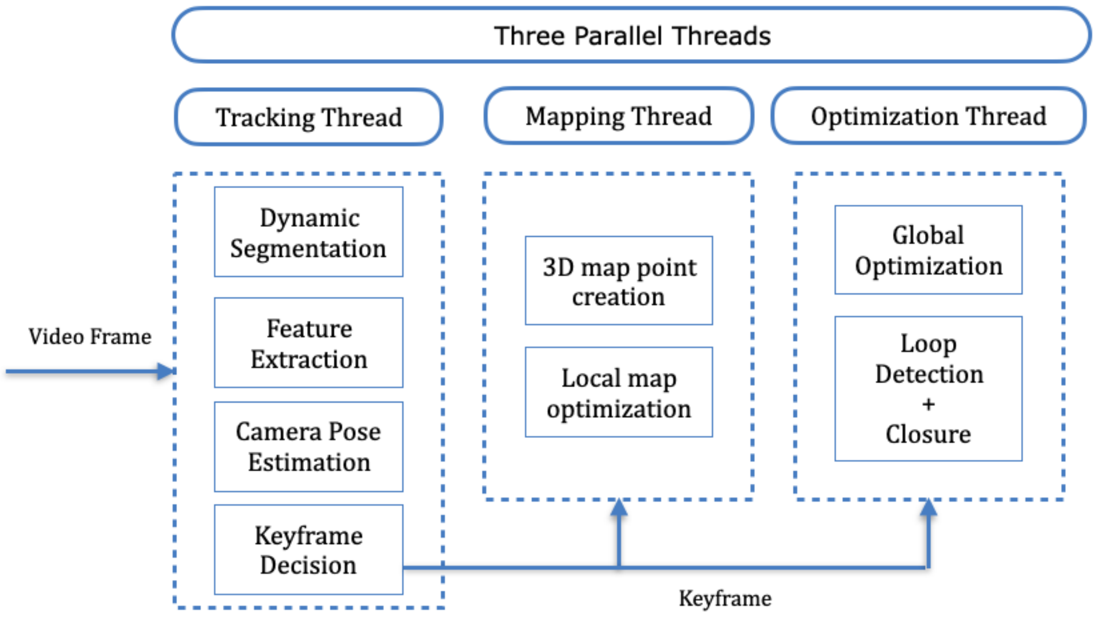


**Figure 2:** Thread Structure in our visual SLAM approach: Tracking thread involves feature extraction, dynamic object detection, followed by camera pose estimation. Mapping thread constructs environment 3D map. Global optimization thread performs overall joint optimization of estimated poses and 3D points

to a previously visited location) in the overall camera trajectory. If any potential loop is detected, this module closes the loop by minimizing the accumulated camera pose error in the loop frame sequence.

## 3.1 Feature Extraction Module

Each incoming frame is passed through a feature extraction module. We have used ORB features [20] (Oriented Multiscale FAST corners [21] and Rotated BRIEF [22]) associated with a 256-bit binary descriptor. ORB features are rotation invariant and have good invariance to viewpoints. But the major advantage of using ORB features is extraction speed as it takes roughly $\sim$ 16 ms for an image size of $640 \times 480$, which is at least one order of magnitude faster compared to other popular feature extractors such as SIFT ($\sim$ 5000 ms) [23], SURF ($\sim$ 300 ms) [24], or the recent A-KAZE ($\sim$ 100 ms) [25].

## 3.2 Potential Dynamic Content Segmentation using CNN-based network

For detecting dynamic objects, we propose to use a CNN-based network that obtains a pixelwise semantic segmentation of the incoming frames. For segmentation classes, we predefine a set of 20 common occurring dynamic class objects - [*person, different kinds of vehicles, some common animals*] and a separate class - *background* for denoting the static objects in the scene. As we are aiming to perform localization in factory environment setup, the dynamic class objects would mostly include humans, vehicles, etc. which has been covered by our predefined set. The semantic segmentation network takes an incoming frame and produces an output frame (same size as input frame) with the pixels detected as belonging to dynamic classes in the predefined set semantically labelled while the rest of the pixels are labelled as background. We further classify all the pixels in this semantically segmented output into two classes - static and dynamic. Therefore, the final output of this segmentation module is a binary mask separating the static and dynamic objects.

In our experiments, we use two popular semantic segmentation networks - Mask RCNN [17] and BiSeNet [26] - and compare their performances in terms of overall accuracy and execution speed of visual SLAM. Mask RCNN can obtain the semantic segmentation and instance labels (although for our purpose we only require semantic segmentation) whereas BiSeNet is only capable of semantic segmentation. The output of the segmentation module, assuming that the input is an RGB image of size $m \times n \times 3$, is a binary mask of size $m \times n$ consisting of the segmentation of detected dynamic objects appearing in the scene. In order to improve the segmentation results and to include the edges of dynamic objects in the mask, we use dilation (with window size= 5) of the segmented binary masks.

## 3.3 Camera Pose Estimation

After the potential dynamic content has been detected and segmented from the frame, the camera pose is tracked using keypoint features from only the static part of the image. Each frame is further passed to the feature extraction module which extracts ORB keypoints. The number of features extracted depends on the image size as well as computational capability of the computing device. For a typical image size of $640 \times 480$, we extract nearly 1000 ORB feature points. For higher image resolution, more number of feature points can be extracted.

For each frame, a prior camera pose is estimated using a constant camera motion model (based on the assumption that the camera movement between current and last frame is same as between the previous two frames). In case IMU data in the form of acceleration is available from the device performing SLAM (such as smartphones, etc.), filtered version of the raw IMU data can be integrated with the prior camera pose for better performance. However, in our case, raw (unfiltered version) IMU data is very noisy and sometimes it caused large deviation in the posterior camera pose. Therefore, we have not used IMU data for estimating camera pose in our experiments. Using the prior camera pose, already existing 3D landmark map points are projected into the frame. These projected points in the current frame are searched for the most suitable nearby ORB feature point. Once a certain number of successful correspondence between nearby ORB features and projected 3D landmark points is obtained, the prior camera pose is optimized using 2D-3D PnP algorithm [27] as shown in Fig. 3. The initially obtained camera pose is optimized again using a set

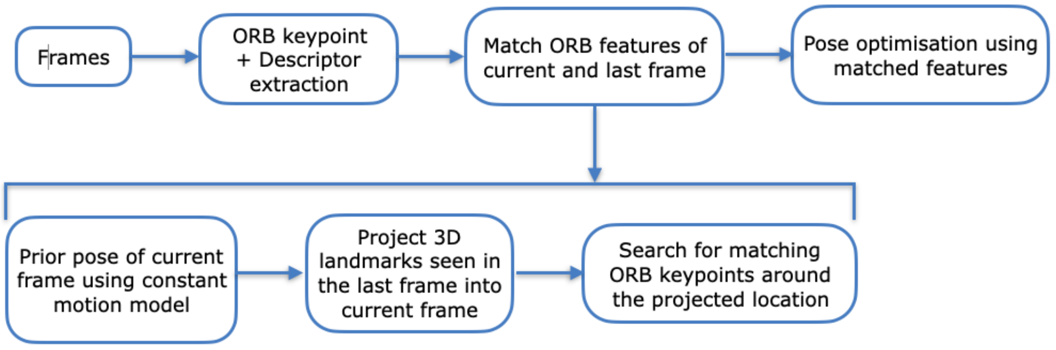

Frames
ORB keypoint + Descriptor extraction
Match ORB features of current and last frame
Pose optimisation using matched features
Prior pose of current frame using constant motion model
Project 3D landmarks seen in the last frame into current frame
Search for matching ORB keypoints around the projected location

**Figure 3:** Major steps for camera pose estimation: ORB features extracted from each incoming frame, 3D map points projected using prior pose and searched for the nearest ORB feature followed by pose optimization using PnP algorithm

of more prominent feature matches between current frame and frames in a local map. The local map includes a set of keyframes $K_1$ which share 3D map point observations with current frame and another set of keyframes $K_2$ connected to keyframe set $K_1$ in the covisibility graph [28] (an undirected weighted graph of all the keyframes as node and edges between them if they share common observations of 3D map points). All the 3D map points observed in this set of local map of keyframes is projected in the current frame using the estimated pose. The same procedure is followed again by finding suitable nearby ORB features in the current frame. Using these extra set of matching 2D features - 3D landmark points, the camera pose is further refined.

The tracking thread is also responsible for deciding whether the current frame qualifies to be a keyframe based on certain conditions mentioned here [6]. Once a frame passes all the conditions to be a keyframe, it is passed on to the mapping and optimization thread which solely process only on keyframes rather than all frames.

## 3.4 Mapping Module

The mapping thread is responsible for processing new keyframes sent to it by the tracking thread and perform local bundle adjustment in order to achieve an optimal 3d map reconstruction in the neighbourhood of current frame location. After receiving the keyframe, it updates the covisibility graph, adding a new node for the current keyframe and updates its graph edges based on common map point observations with other keyframes. The mapping thread estimates new map points (if any, in the current keyframe) by triangulating ORB keypoints from connected keyframes in the covisibility graph as shown in Fig. 5. There are situations when wrong 3D points are triangulated due to feature matching errors and wrong feature associations. In order to ensure that newly triangulated 3D points are trackable, the mapping thread imposes certain restrictive conditions on these newly created points. If and when a certain 3D point fails to fulfill these conditions, they are rejected from the map. The local bundle adjustment optimizes the currently processed

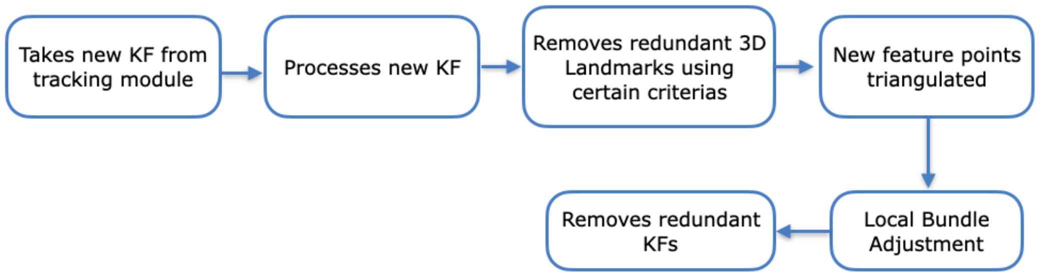

Takes new KF from tracking module
Processes new KF
Removes redundant 3D Landmarks using certain criterias
New feature points triangulated
Local Bundle Adjustment
Removes redundant KFs

**Figure 4:** Major steps in the mapping module: processes only keyframes, estimates new undiscovered 3D map point in the scene, removes redundant untrackable map points and keyframes, and optimizes poses and map points of keyframes in local environment to current keyframe

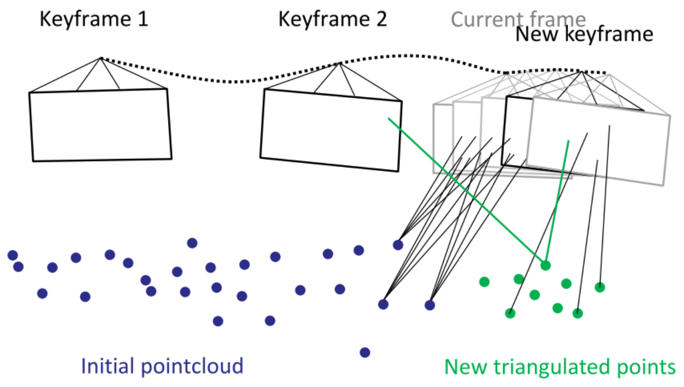


**Figure 5:** Figure demonstrating a set of keyframes and creation of new map points visible and undiscovered in the current keyframe

keyframe, all the poses of the keyframes connected to it in the covisibility graph, and all the map points seen by those keyframes. All other keyframes that see those points but are not connected to the currently processed keyframe are included in the optimization but remain fixed. In order to maintain a compact map representation, this thread tries to detect redundant keyframes and remove them. This is beneficial as bundle adjustment complexity grows with the number of keyframes added to the map, but also because it enables lifelong operation in the same environment as the number of keyframes will not grow unbounded, unless the visual content in the scene changes. All the major steps performed by this thread have been shown in a sequential manner in Fig. 4.

## 3.5 Loop Detection and Closure

A loop in this context refers to trajectory camera sequence where the starting and end points of the sequence belong to the same environment. The optimization module first needs to search for any potential loops with every incoming keyframe and if detected, close them by minimizing the accumulated pose drift error along the loop. Briefly describing the loop detection and closing process as portrayed in Fig. 6, firstly, each incoming keyframe is converted to bag-of-words (BoW) vector [29]. Using this BoW vector, the system searches for the most similar keyframe to current keyframe by comparing the BoW vector of all the keyframes in the keyframe database. If a similar candidate match is found in the database, it is considered as a loop candidate for the current keyframe. Once loop detection is done, the loop needs to be closed as it provides an opportunity to mitigate the accumulated pose drift error along the loop trajectory sequence. In order to close the loop, the graph of camera poses from current to loop candidate keyframe is optimized by computing a similarity transformation between consecutive keyframes in the loop and minimizing the residual error between consecutive poses with the objective of distributing this error along the graph as shown in Fig. 7.

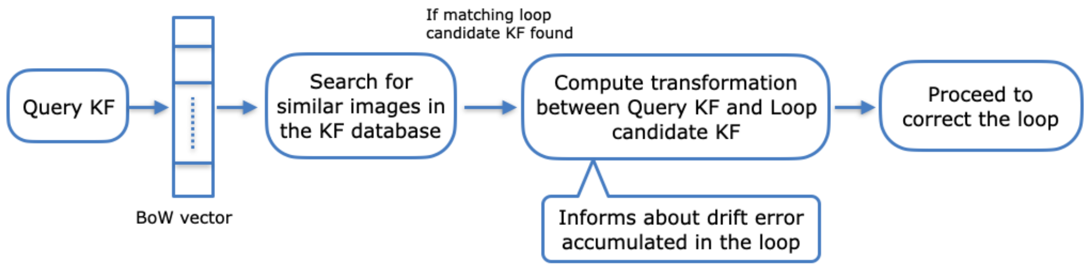

Query KF
BoW vector
Search for similar images in the KF database
If matching loop candidate KF found
Compute transformation between Query KF and Loop candidate KF
Informs about drift error accumulated in the loop
Proceed to correct the loop

**Figure 6:** Flowgraph of the loop detection and closure process: each incoming keyframe is converted to BoW vector using which the most similar keyframe in the database is searched. If found, the loop is closed minimizing the accumulated drift error along the loop camera poses

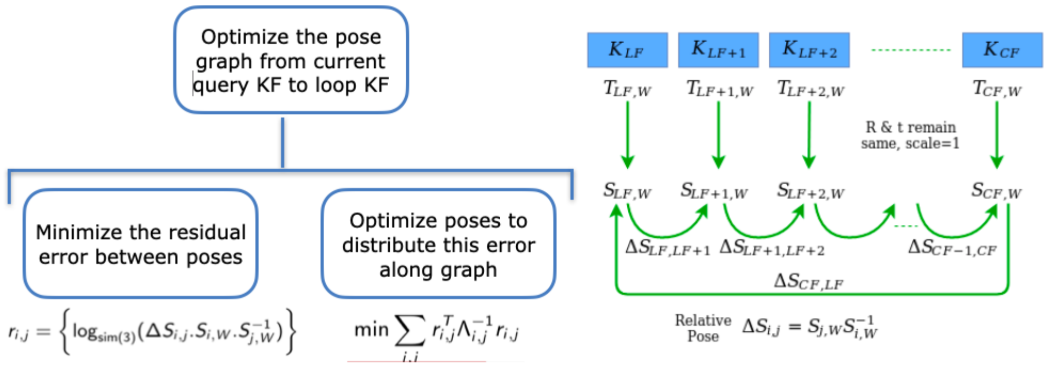

Optimize the pose graph from current query KF to loop KF
Minimize the residual error between poses
Optimize poses to distribute this error along graph
$r_{i,j} = \left\{ \log_{sim(3)}(\Delta S_{i,j} . S_{i,W} . S_{j,W}^{-1}) \right\}$
$\min \sum_{i,j} r_{i,j}^T \Lambda_{i,j}^{-1} r_{i,j}$
$K_{LF}$ $K_{LF+1}$ $K_{LF+2}$ $K_{CF}$
$T_{LF,W}$ $T_{LF+1,W}$ $T_{LF+2,W}$ $T_{CF,W}$
R & t remain same, scale=1
$S_{LF,W}$ $S_{LF+1,W}$ $S_{LF+2,W}$ $S_{CF,W}$
$\Delta S_{LF,LF+1}$ $\Delta S_{LF+1,LF+2}$ $\Delta S_{CF-1,CF}$
$\Delta S_{CF,LF}$
Relative Pose $\Delta S_{i,j} = S_{j,W} S_{i,W}^{-1}$

**Figure 7: Left:** Residual error between camera poses i and j defined as similarity transformation as well as the minimizing function using the residual error. **Right:** demonstration of computing similarity transformation along the loop from current keyframe $K_{CF}$ to loop candidate keyframe $K_{LF}$

Finally, the entire research method combining all the three modules - tracking, mapping and optimization - that we propose to use for our final smartphone application has been presented in Fig. 8. In this figure, we have included the idea of using GPS information available from smartphone for getting the coarse location of the device performing SLAM which can be used for automatically detecting the correct stored map when the user starts the application.

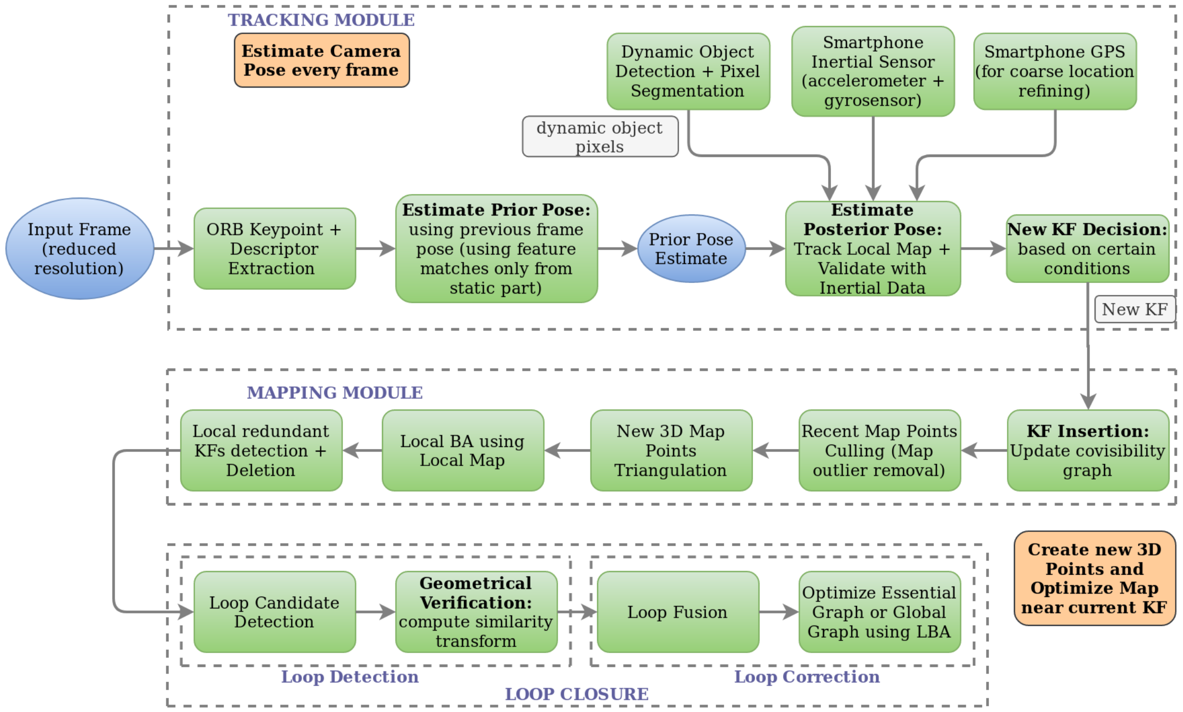

TRACKING MODULE
Estimate Camera Pose every frame
Dynamic Object Detection + Pixel Segmentation
Smartphone Inertial Sensor (accelerometer + gyrosensor)
Smartphone GPS (for coarse location refining)
dynamic object pixels
Input Frame (reduced resolution)
ORB Keypoint + Descriptor Extraction
Estimate Prior Pose: using previous frame pose (using feature matches only from static part)
Prior Pose Estimate
Estimate Posterior Pose: Track Local Map + Validate with Inertial Data
New KF Decision: based on certain conditions
New KF
MAPPING MODULE
Local redundant KFs detection + Deletion
Local BA using Local Map
New 3D Map Points Triangulation
Recent Map Points Culling (Map outlier removal)
KF Insertion: Update covisibility graph
Create new 3D Points and Optimize Map near current KF
Loop Candidate Detection
Geometrical Verification: compute similarity transform
Loop Fusion
Optimize Essential Graph or Global Graph using LBA
Loop Detection
Loop Correction
LOOP CLOSURE

**Figure 8:** Block diagram of our entire visual-inertial SLAM module for localization of human workers: Three threads - tracking, mapping, optimization.

## 3.6 Re-Localization and Navigation

Re-Localization refers to a situation when the tracking module is unable to estimate the current camera pose. This can happen due to various factors - 1. texture-less scenes (such as plain walls) leading to small number of features available for matching and optimization 2. abrupt and sudden camera movement can cause little to no features common between current and previous frames as well as local keyframes resulting in inaccurate matching. Thus, in these cases, the normal way of estimating camera pose does not works and the system has to perform place recognition (similar process to loop detection) for finding the camera pose.

Our main objective in this research is navigation of human workers in order to help them to get from one place to another in a factory. This requires a pre-built map of the environment in which the user can localize himself as well as navigate to their target location. The pre-built map of one particular area or factory is a one-time initial setup activity and has to be created in advance using the visual-inertial SLAM technique discussed in above sections and demonstrated in Fig. 8. When the user wants to localize himself or get navigational help, they can manually select the stored map corresponding to that factory-setup or this can automatically be done using the GPS location of the user. Once the user starts to move, the system will initially perform place recognition by finding the most similar keyframe in the map keyframe database to the current frame by comparing their BoW vectors (similar to loop detection process described in Fig. 6). The initial time taken for matching depends on the map size and also on whether the query location or scene has been visited in the map or not. A successful matching and localization for a map-size of ~ 1200 keyframes takes nearly 4-5 seconds. After an initial match has been found using place recognition process, the system will continue tracking the person using just the tracking module and optimization module in Fig. 8. It is important to note that the mapping module is not required in the scenario of localization and navigation as the map is already available.

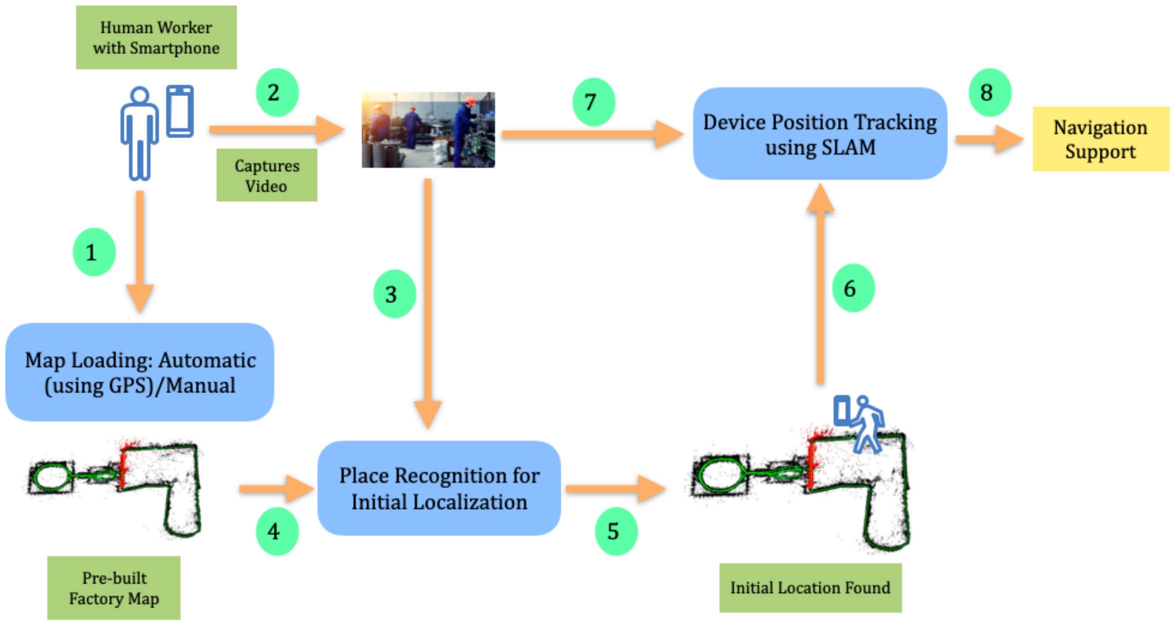

Human Worker with Smartphone
2
Captures Video
7
Device Position Tracking using SLAM
8
Navigation Support
1
3
6
Map Loading: Automatic (using GPS)/Manual
Place Recognition for Initial Localization
4
5
Pre-built Factory Map
Initial Location Found

**Figure 9:** Demonstration of the self-localization application for human workers

## 4 Evaluation Metrics

A SLAM system generally outputs the estimated camera trajectory along with an estimate of the resulting map. While it is in principle possible to evaluate the quality of the resulting map, accurate ground truth maps are difficult to obtain. Therefore, the quantitative performance of any SLAM-based system is evaluated by the quality of the estimated trajectory. It should be noted that a good trajectory does not necessarily imply a good map, as for example even a small error in the map could prevent the system from functioning properly in the environment and may also lead to tracking failure.

For the evaluation of SLAM tracking and localization, we use two common evaluation metrics introduced by Engel et. al in [30]. We assume the sequence of poses obtained from the estimated trajectory as $P_1, ...., P_n \in SE(3)$ and those obtained from the ground truth trajectory as $Q_1, ... , Q_n \in SE(3)$

1. **Relative Pose Error (RPE)** - RPE measures the local accuracy of the trajectory over a fixed time interval $\Delta$. In our experiments, we have considered $\Delta = 1$ as the intuitive choice because consecutive frames are being matched for estimating camera pose. Therefore, this metric corresponds to the drift of the trajectory useful for the evaluation of visual SLAM systems. Relative Pose Error at any time step $i$ is defined as
$$E_i = \left(Q_i^{-1} Q_{i+\Delta}\right)^{-1} \left(P_i^{-1} P_{i+\Delta}\right) \tag{1}$$
From these errors, the root mean squared error (RMSE) of the translational component of $E_i$ over all time indices $i$ is given by -
$$\text{RMSE}(E_{1:n}, \Delta) = \left(\frac{1}{m}\sum_{i=1}^{m} \|\text{trans}(E_i)\|^2\right)^{1/2} \tag{2}$$

2. **Absolute Pose Error (APE)** - This metric measures the global consistency of the estimated trajectory i.e, it compares the absolute distance between the estimated and ground-truth trajectory. As both trajectories can be specified in arbitrary coordinate frames, they first need to be aligned. This can be achieved by using Umeyama Alignment [46], which finds the rigid-body transformation $S$ corresponding to the least-squares solution that maps the estimated trajectory $P_{1:n}$ onto the ground truth

trajectory $Q_{1:n}$. The absolute trajectory error at any time step $i$ is computed as

$$F_i = Q_i^{-1} S P_i \tag{3}$$

Similar to RPE, the RMSE of the translational component is given by

$$\mathrm{RMSE}(F_{1:n}) = \left( \frac{1}{n} \sum_{i=1}^{n} \|\mathrm{trans}(F_i)\|^2 \right)^{1/2} \tag{4}$$

The RPE considers both translational and rotational errors, while the ATE only considers the translational errors. RPE metric provides an elegant way to combine rotational and translational errors into a single measure. However, rotational errors typically also manifest themselves in wrong translations and are thus indirectly also captured by the ATE. the two metrics are strongly correlated.

## 5 Datasets

There exists a number of datasets for quantitatively evaluating the performance of visual SLAM and localization algorithms. We have conducted experiments and performance comparison using two robust datasets - TUM RGB-D Dataset [31] and KITTI Dataset [32]. Until now in this research, we have not performed any quantitative evaluation with our target area (factory related environment) because of lack of proper dataset as well as corresponding ground-truth information but we plan to work on this in the future.

### 5.1 TUM RGB-D Dataset

This dataset consists of 39 RGB-D image sequences (with 6-DoF camera poses) and near accurate ground-truth trajectory information recorded in various indoor environments. These sequences are recorded using Microsoft Kinect sensor at a frame rate of 30 Hz. Most image sequences in this dataset are completely static-content based environment with only 9 sequences having some dynamic content in the scenes. In this research, we focus on evaluating the performance of algorithms on these nine image sequence scenes involving dynamic contents. Both static-based and dynamic-based SLAM algorithms are evaluated on these sequences and compared with additional variations which are explained later.

**Brief Description of the dynamic sequences:** In the sequences named *sitting*, there are two people sitting while speaking and gesticulating, i.e., there is low degree of motion. In the sequences named *walking* (w), two people walk as shown in Fig. 10. For both types of sequences *sitting* (s) and *walking* (w) there are further four types of camera motions: (1) halfsphere (half): the camera moves following the trajectory of a half sphere, (2) xyz: the camera moves along the x-y-z axes, (3) rpy: the camera rotates over roll, pitch and yaw axes, and (4) static: the camera is manually kept static.

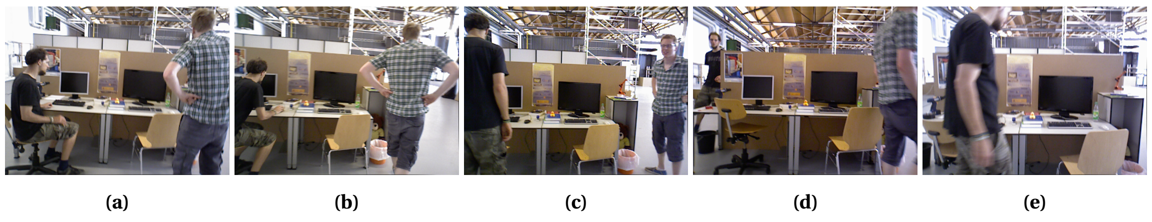


**Figure 10:** Raw example images taken from one of the walking sequence of TUM RGB-D dataset showing people walking with camera movement

### 5.2 KITTI Dataset

The KITTI Dataset consists of 22 stereo sequences recorded at a frame rate of 10 Hz from a car in urban and highway environments with a mixed content of static and dynamic objects in each sequence. In general, the static objects in the scenes include houses, trees, bushes, etc. whereas the dynamic objects mainly consists of various types of vehicles, people, etc. as shown in Fig. 11. We use only 11 sequences out of total 22, as only they have ground-truth information available.

**Figure 11:** Raw example images taken from various sequences of KITTI dataset showing outdoor environment scenes consisting of static and dynamic objects captured using a camera mounted on top of moving vehicle

## 6 Experiments and Results

The experimental operating environment is a CentOS-based desktop computer configured as an Intel Core i7-7300HQ CPU (4-core 2.5 GHz), 32 GB of RAM, and NVIDIA Titan X (Pascal) with 6 GB of graphic memory for semantic segmentation module. For each meaningful evaluation of an algorithm on an image sequence, we run it multiple times (3 or 5 times) and take the median of the error metric value in order to mitigate the effects of randomness and uncertainties.

### 6.1 Dynamic object segmentation module - Mask RCNN

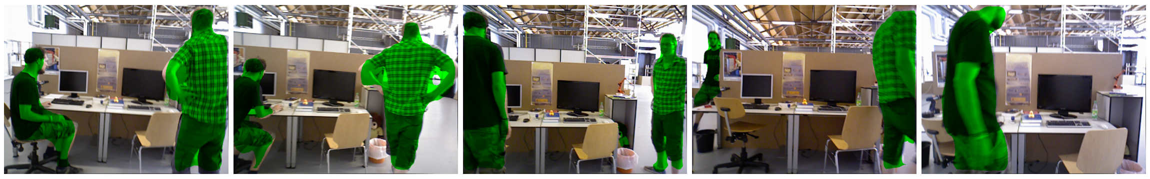
In this research, we use Mask RCNN network for segmentation of dynamic class objects. We use the Mask RCNN pre-trained on MS COCO dataset [33] for instance-level segmentation task implemented by MatterPort [34]. Each incoming video frame is passed through Mask RCNN network trained to detect and segment objects belonging to 80 different classes. Out of these 80 classes, we only consider 20 manually selected dynamic class categories such as human, various common occuring vehicles (car, bus, bicycle, etc.), and some common animals. The Mask RCNN network outputs a binary mask (of the same size as input frame), segmenting the input frame into static and dynamic content. Fig. 12 and Fig. 14 shows the segmentation output of Mask RCNN overlayed on the input frames from TUM RGB-D and KITTI dataset respectively. This segmented mask information is forwarded to the visual SLAM module which performs further computation using extracted ORB keypoint features taken only from the static image region and rejects the keypoints belonging to the detected dynamic objects.

When Mask RCNN network is integrated with the SLAM module, the overall camera pose estimation (tracking) speed is roughly around 6-7 fps. Thus, the Mask RCNN creates a bottleneck in the entire system as the segmentation speed of Mask RCNN is roughly around 5-6 fps on an NVIDIA Tesla M40 GPU as reported by He et al. [17]. In order to improve the tracking speed so that the system can run in real-time, we tried different variants of segmentation approach by skipping dynamic segmentation on each and every frame and pass every $m^{\text{th}}$ frame to Mask RCNN network. For the intermediate frames, for which the dynamic segmentation is not performed explicitly by the network, we use segmentation output from the immediate last segmented frame. In this experiment, we use $m = 1$ (every frame), $m = 2$ (every other frame), $m = 5$, $m = 10$ and compare the performance of these variants by computing the RPE and APE as described in section 4.

#### 6.1.1 TUM RGB-D dataset

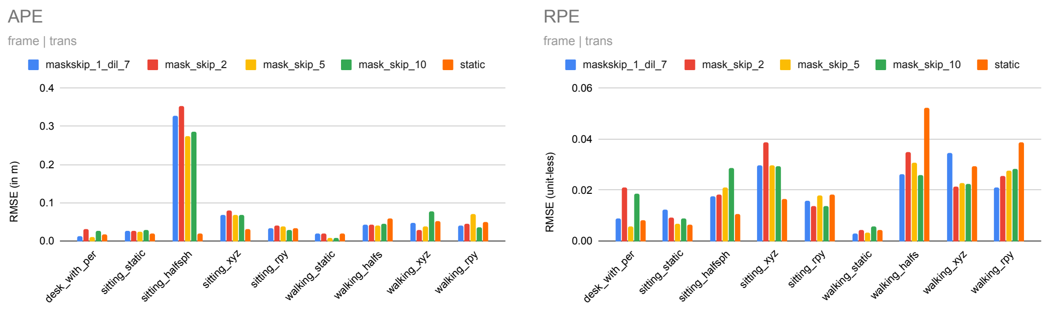
Table 1: TUM RGB-D dataset: Average RMSE taken over all the sequences for each of the five variants compared in Fig. 13

| Error Metric | mask skip 1 | mask skip 2 | mask skip 5 | mask skip 10 | static |
|---|---|---|---|---|---|
| APE | 0.06893 | 0.07374 | 0.06368 | 0.06722 | **0.03345** |
| RPE | 0.01874 | 0.02085 | **0.01839** | 0.02017 | 0.02054 |



(a) (b) (c) (d) (e)

**Figure 12:** Detection and segmentation of dynamic class objects in scenes taken from the *walking* sequence of TUM RGB-D dataset using Mask RCNN network. In these example images, there is only a single dynamic class object in the scene - human, and the network robustly detects with good acceptable accuracy



(a) (b)

**Figure 13:** TUM RGB-D dataset: Comparison of (a) APE and (b) RPE for different variants of dynamic segmentation approach (with mask-skip parameter $m = 1, 2, 5, 10$) and static-based approach (without dynamic segmentation)

**Observations:** Fig. 13 shows the evaluation results comparison of APE and RPE for different variants of Mask RCNN based SLAM and static based SLAM on the dynamic sequences from TUM RGB-D dataset. Table 1 summarizes the average RMSE error taken over the nine dynamic sequences from the TUM RGB-D dataset for each of the five variants compared in Fig. 13. Following are the observations and reasoning based on these results:

1. The segmentation result of Mask RCNN network in the images from this dataset are pretty accurate. The implementation of Mask RCNN network was directly adapted from MatterPort [34], without any requirement of fine-tuning. In order to mitigate boundary effects (improper segmentation of boundary pixels belonging to a dynamic class object can lead to erroneous ORB features), we use dilation of binary masks (with a small window size=5). In some frames, the network randomly detects small blobs of static objects as belonging to dynamic classes which might bring some inaccuracy to the overall performance (almost negligible).

2. The performance of static-based approach is nearly comparable or better than Mask RCNN based dynamic segmentation approach in *sitting* sequences (very less dynamic content). One possible explanation for this behavior can be due to very less dynamic content in these sequences (described in Sec. 5.1), as the two persons are speaking and gesticulating while sitting. This will lead to rejection of potential feature points belonging to dynamic objects that could have been used for matching as those dynamic objects are not moving.

3. The dynamic segmentation approach performs better or comparable to the static-based approach in *walking* sequences (with prominent dynamic content). In this set of sequences, as the dynamic class objects are moving and walking around, the feature points belonging to the moving people should not be taken into account for any of the feature matching tasks in the visual SLAM pipeline. This represents the ideal way of estimating camera pose with reduction in erroneous feature points. This is done by the dynamic segmentation approach contrary to the static-based approach which considers all the feature points available in the scene.

4. Based on the results and corresponding reasoning described in point 2 and 3, we observe that if an image sequence has dynamic class objects which are static for most part of the sequence, then the segmentation approach will cause rejection of feature points belonging to those non-moving dynamic class objects. Thus, in these cases, the dynamic class objects apparently behave similar to static objects and their dynamic nature does not affects the performance of static-scene based visual SLAM to a great extent. Hence, in the *sitting* sequences, the static-based approach produces lower error compared to dynamic segmentation approach.

5. Among the various mask-skip variants $m = 1, 2, 5, 10$, the results are ambiguous and does not present any definite conclusion as can be seen from the Fig. 13 as well as Table 1. But based purely on logical evidence, we can say that performing dynamic segmentation for every consecutive frame ($m = 1$) would be more accurate than skipping frames. The expected result in these mask-skip variants should be increase in performance error as we increase the value of mask-skip parameter $m$. The reason for using mask-skip variants was to improve segmentation speed and thus the overall running speed, while keeping the performance acceptable.

### 6.1.2 KITTI dataset

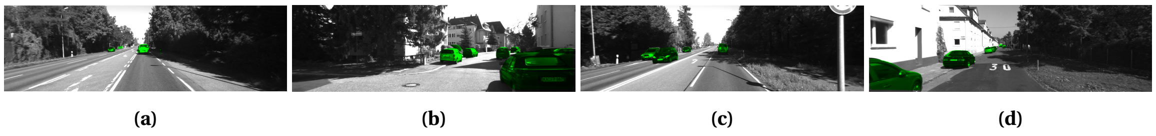


**Figure 14:** Detection and segmentation of dynamic class objects in the scenes from KITTI dataset using Mask RCNN network

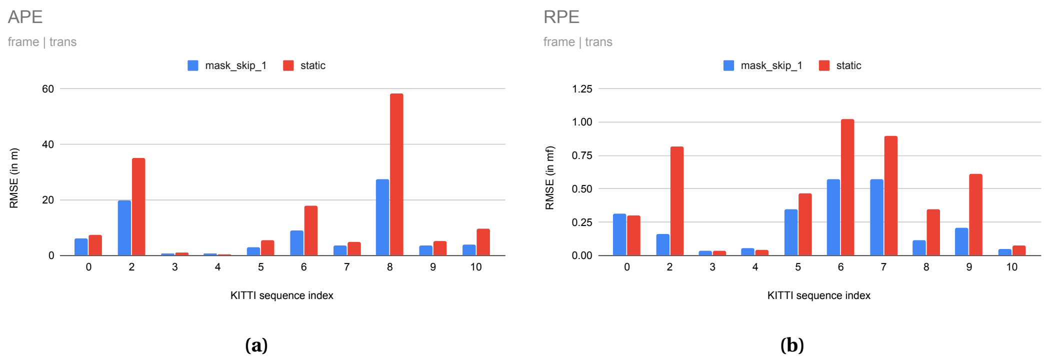


**Figure 15:** KITTI dataset. Comparison of (a) APE and (b) RPE for dynamic segmentation approach (only with $m = 1$) and static-based approach (without dynamic segmentation)

Table 2: KITTI dataset: Average RMSE taken over all the sequences for dynamic segmentation approach and static-based approach compared in Fig. 15

| Error Metric | mask skip 1 | static |
|---|---|---|
| APE | **7.8108** | 14.5872 |
| RPE | **0.2437** | 0.4614 |

**Observations:** Fig. 15 shows the evaluation results comparison of APE and RPE for dynamic segmentation approach and static-based approach on the KITTI dataset. Based on these results, following observations can be made:

1. In this dataset, the segmentation results of Mask RCNN network is very much acceptable without any fine-tuning as evident in Fig. 14. The dynamic class objects in this dataset sequence mostly include cars, vehicles, bicycles, and human.
2. The performance of dynamic segmentation approach is better than static-based approach in nearly all the sequences of this dataset except sequence 0 and sequence 4 as can be seen in Fig. 15. This dataset involves large camera movements captured from moving car as well as there is good amount of dynamic content in the scene as compared to *sitting* sequences from TUM RGB-D dataset. Therefore the results in these sequences between the two approaches are more contrasting compared to TUM RGB-D dataset.
3. We compare the average RMSE taken over all the sequences for both the approaches as shown in Table 2. From this table, it is evident that the dynamic approach is performing better compared to static-based approach for this dataset.
4. We did not perform experiments with mask-skip approach for this dataset sequence, as the camera movement between frames is significant enough. This was resulting in change in position of dynamic objects between frames which are separated by 5 or 10 frames. Thus, using masks from past frames for current frame resulted in mis-placement of segmented mask.

#### 6.1.3 Experimental Conclusion

1. Dynamic segmentation approach performs better than static-based approach in real-world scenarios with dynamic content. In static-scene environment, where dynamic scene content is small or negligible, the performance of dynamic segmentation approach is nearly comparable and not significantly reduced or affected compared to static-based approach.
2. In all cases and scenarios, the dynamic segmentation approach has one basic advantage over static-based approach. The environment map created in the former case does not consists of 3D landmark points belonging to dynamic objects appearing along the sequence. This leads to creation of re-usable static maps consisting of map-points only from structural objects and therefore can be re-used for long-term applications.
3. In our application scenario of localization of human workers in factories, we need to create a static re-usable map of the environment by removing the effects of dynamic class objects in the scene. Even if the dynamic class object is static and not moving during map creation, it can move later and will lead to inaccuracies during localization when the user is navigating. In case the map is created without any dynamic objects in the environment (one-time activity so this can be done), during localization, the dynamic moving objects will pose a problem and hence need to be removed from the scene using segmentation approach.

## 7 Problems encountered and bottlenecks

1. We faced issues while integrating the visual SLAM module (written in native C/C++ for fast real-time performance) with dynamic segmentation module (written in python). The bridging involved conversion and sending of image data from C++ Mat object to python object data and vice-versa. This problem has been resolved and we currently have a successful bridge for dynamic segmentation based visual SLAM.
2. Currently, we are working on the creation of the smartphone application for self-localization. One major problem encountered was the integration of various 3rd party libraries in the android project which took a lot of effort and time. The integration has been done for all the libraries except one - pangolin used for map rendering and visualization. We are looking for an alternative approach to this.
3. The speed of the Mask RCNN segmentation in the SLAM pipeline is a major bottleneck since we need to perform in real-time. One alternative is to use mask-skip approach with mask-skip value $m = 2$ (leaving every other frame) leading to improvement in speed.

## 8 Safety Considerations

This project has potential application in the safety of human workers in factories. For potential risk detection involving dangerous or heavy objects, the basic principle is to estimate the positional relationship between any human worker and the dangerous object. Our current in-progress localization system can be used to detect workers location inside factories, which can be compared with dangerous object's position to determine whether the scenario poses any risk to the worker. In this way, the self-localization technology can be used for improving safety of workers in factories.

## 9 Environmental issues considered

This research does not include any activity that impacts the environment in any significant way.

## 10 Benchmark of this research

Research and development in the field of visual-inertial SLAM has been going on for a while now and many companies have been using and advancing this technology according to their needs. Apple released its own AR technology called "ARKit" which is a software development kit providing functionalities for device motion tracking and mapping. This tool can only be used for apple-based devices. DragonFly by Accuware has developed SLAM-based visual positioning system that provides indoor location to robots, drones, forklifts and other vehicles, in GPS-denied environments, using a standard camera. Another SLAM technology called "GrandSLAM" has been developed by Kudan, which accepts a wide range of sensor data such as camera, ToF, Lidar, IMU, etc. for device pose estimation. One of the examples of their applications is the localization software for autonomous vehicles, robo taxis, etc.

These competitors have made progresses both in technology development and commercial availability. DragonFly and Kudan's GrandSLAM does not provide any description of their algorithm. Since there is no description about their method, it may be reasonable to guess that theirs are not designed especially to deal with dynamic objects in real-world environments as it is not mentioned anywhere in their description.

## 11 Consideration of IPR status

### 11.1 Intellectual property owned by other organisations (domestic/foreign patent)

Table 3 provides brief details about a few of the patent filed on the technology of pose estimation using SLAM

## 12 Conclusion

### 12.1 Summary

In this research, we have integrated a visual SLAM system (building on OpenVSLAM) with a dynamic object segmentation module capable of detecting and segmenting dynamic objects in the scene. This approach makes the overall system robust in situations where the dynamic content is large and represents an important part of the scene as well as there is sufficiently large camera motion. Our approach also creates a static-object based and therefore reusable map of the environment. The estimated map only contains structural objects and can therefore be re-used in long-term applications.

### 12.2 Future issues

In this work, we have done few experiments focused on improving the performance of SLAM in dynamic environments and shown the results using standard datasets. Next, we are working to create an android-

Table 3: Existing patents related to the technology of localization and pose estimation using SLAM

| NO. | Applicant | Publication no. | Filing date |
|---|---|---|---|
| 1 | Starship Technologies/Ahti Heinla et al. | US20190354110A1 | 2019-07-30 |
| | Name of the invention: | Mobile robot system and method for autonomous localization using straight lines extracted from visual images | |
| | Brief explanation of the invention: | This patent presents a mobile robot which localizes itself by extracting features from visual images and comparing them with existing map data | |

| NO. | Applicant | Publication no. | Filing date |
|---|---|---|---|
| 2 | Starship Technologies/Ahti Heinla et al. | JP2018532204A | 2016-11-02 |
| | Name of the invention: | Device and method for autonomous self-location estimation | |
| | Brief explanation of the invention: | This patent presents a mobile robot which localizes itself by extracting features from visual images and comparing them with existing map data | |

| NO. | Applicant | Publication no. | Filing date |
|---|---|---|---|
| 2 | Huawei Technologies Co., Ltd/Panji Setiawan et al. | WO2020048623A1 | 2020-03-12 |
| | Name of the invention: | Estimation of a pose of a robot | |
| | Brief explanation of the invention: | This patent presents a method for estimating robot pose using a feature-based method. The camera pose is estimated using a successive weighted sum of previous pose estimates. The weights are chosen using improved particle filtering method. | |

based smartphone application that can perform self-localization and add navigational capabilities to the application for human workers in factories.

## References

[1] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, “Monoslam: Real-time single camera slam,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[2] J. Civera, A. J. Davison, and J. M. M. Montiel, “Inverse depth parametrization for monocular slam,” *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 932–945, 2008.

[3] H. Strasdat, J. M. M. Montiel, and A. J. Davison, “Real-time monocular slam: Why filter?” In *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 2657–2664.

[4] G. Klein and D. Murray, “Parallel tracking and mapping for small ar workspaces,” in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007, pp. 225–234.

[5] J. Engel, T. Schöps, and D. Cremers, “Lsd-slam: Large-scale direct monocular slam,” in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Cham: Springer International Publishing, 2014, pp. 834–849.

[6] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, “Orb-slam: A versatile and accurate monocular slam system,” *IEEE Transactions on Robotics*, vol. 31, no. 5, 1147–1163, 2015.

[7] B.-f. Chen and Z. Cai, “Research on mobile robot slam in dynamic environment,” *Applied Mechanics and Materials*, vol. 130-134, Oct. 2011.

[8] K. Pirker, M. Rüther, and H. Bischof, “Cd slam - continuous localization and mapping in a dynamic world,” in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 3990–3997.

[9] L. Xiao, J. Wang, X. Qiu, Z. Rong, and X. Zou, “Dynamic-slam: Semantic monocular visual localization and mapping based on deep learning in dynamic environment,” *Robotics and Autonomous Systems*, vol. 117, pp. 1 –16, 2019.

[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” *Lecture Notes in Computer Science*, 21–37, 2016.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, *You only look once: Unified, real-time object detection*, 2015. arXiv: `1506.02640 [cs.CV]`.

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik, *Rich feature hierarchies for accurate object detection and semantic segmentation*, 2013. arXiv: `1311.2524 [cs.CV]`.

[13] K. He, X. Zhang, S. Ren, and J. Sun, *Spatial pyramid pooling in deep convolutional networks for visual recognition*, 2014. arXiv: `1406.4729 [cs.CV]`.

[14] R. Girshick, *Fast r-cnn*, 2015. arXiv: `1504.08083 [cs.CV]`.

[15] S. Ren, K. He, R. Girshick, and J. Sun, *Faster r-cnn: Towards real-time object detection with region proposal networks*, 2015. arXiv: `1506.01497 [cs.CV]`.

[16] J. Long, E. Shelhamer, and T. Darrell, *Fully convolutional networks for semantic segmentation*, 2014. arXiv: `1411.4038 [cs.CV]`.

[17] K. He, G. Gkioxari, P. Dollár, and R. Girshick, *Mask r-cnn*, 2017. arXiv: `1703.06870 [cs.CV]`.

[18] S. Sumikura, M. Shibuya, and K. Sakurada, “Openvslam,” *Proceedings of the 27th ACM International Conference on Multimedia*, 2019.

[19] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, “Bundle adjustment - a modern synthesis,” in *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ser. ICCV ’99, Berlin, Heidelberg: Springer-Verlag, 1999, 298–372, ISBN: 3540679731.

[20] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *2011 International Conference on Computer Vision*, 2011, pp. 2564–2571.

[21] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 430–443, ISBN: 978-3-540-33833-8.

[22] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “Brief: Binary robust independent elementary features,” in *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos, and N. Paragios, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 778–792, ISBN: 978-3-642-15561-1.

[23] D. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, pp. 91–, Nov. 2004.

[24] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417, ISBN: 978-3-540-33833-8.

[25] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, “Kaze features,” in *Computer Vision – ECCV 2012*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 214–227, ISBN: 978-3-642-33783-3.

[26] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, *Bisenet: Bilateral segmentation network for real-time semantic segmentation*, 2018. arXiv: `1808.00897 [cs.CV]`.

[27] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal of Computer Vision*, vol. 81, Feb. 2009.

[28] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige, "Double window optimisation for constant time visual slam," in *2011 International Conference on Computer Vision*, 2011, pp. 2352–2359.

[29] D. Galvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[30] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 573–580.

[31] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, 2012.

[32] A Geiger, P Lenz, C Stiller, and R Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[33] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, *Microsoft coco: Common objects in context*, 2014. arXiv: `1405.0312 [cs.CV]`.

[34] Matterport, *Mask rcnn*, `https://github.com/matterport/Mask_RCNN`, 2013.