A Continuous Monocular Visual Odometry Pipeline

Maximilian Frühauf*, Pascal Roth†, Albert Gassol Puigjaner‡, Adrià López Escoriza§
ETH Zürich
*21-944-707, †21-951-199, ‡21-947-445, §20-908-034

***Abstract*—This report presents a monocular continuous Visual Odometry (VO) pipeline designed for the purpose of vehicle localization. The work has been carried out as the mini-project for the Visual Algorithms for Mobile Robotics[1] course at the University of Zurich. Evaluation has been performed on three different datasets and as result, it is argued that the proposed pipeline (i) achieves local consistency in the estimated poses (ii) optimized through a continuous bundle adjustment approach, and (iii) can be further refined in terms of robustness and accuracy by employing inertial measurements.**

I. Introduction

Visual Odometry (VO) is the process of estimating the rigid body transformation of an object, given an ordered sequence of images. Since the 1980s several robust methods have been developed and employed in a great range of applications in different fields. Particularly in robotics, they have revolutionized tracking and mapping algorithms by incorporating visual measurements into the robot state estimator and later combining them with motion sensors such as Inertial Measurement Unit (IMU) and other information sources, e.g. Global Positioning System (GPS). In this project, we first demonstrate a fully vision-based pose estimation system using a monocular camera. In a second instance, additional high-frequency IMU measurements are included in the pipeline to refine the estimations and increase robustness, resulting an a so-called Visual Inertial Odometry (VIO) pipeline.

A. Workflow

General concepts of a VO and VIO pipeline have been taken from the lecture *Visual Algorithms for Mobile Robotics*. Given the constraints of the project, a four-stage timeline has been established:

1) **Design & Concept**: Determining the overall pipeline structure and a basic code skeleton as well as investigating different additional features to distinguish the given work from the minimum requirements.
2) **Implementation**: Constructing the different VO building blocks in a way that combination in a plug-and-play manner is possible and evaluating different available methods. For instance, a comparison has been carried out between Kanade-Lucas-Tomasi (KLT) tracking performance and classical feature matching.
3) **Integration**: Combing the building blocks into a fully functional pipeline while further evaluating different performances.
4) **Evaluation**: Testing on Parking, KITTI and Malaga datasets while simultaneously optimizing parameter settings to adjust for the differences and increase performance. Gathering results and comparing them to the outlined goals.

[1] http://rpg.ifi.uzh.ch/teaching.html

B. Datasets

The evaluation datasets have been recorded in different urban environments with a variation in movements, weather conditions, and general surroundings. In the following, each set is presented shortly.

1) **Parking**: In a simulation, images are taken from a vehicle following a straight trajectory inside a parking garage. The vision setup consists of a single camera stream at 20 Frames per Second (FPS). Due to the simulated environment, ground-truth poses are available.
2) **KITTI**: Images are recorded by a vehicle that is equipped with a stereo camera and an IMU. However, this work only uses a single camera view and. Moving in a suburban environment in the presence of dynamic objects such as other vehicles and pedestrians provides an increased difficulty compared to the Parking set [1]. In addition, ground truth position data is also available.
3) **KITTI IMU**: Same dataset as **KITTI** with high frequency IMU measurements as well as transformations from IMU to vehicle and camera reference frame.
4) **Malaga**: Images are recorded by a vehicle driving in an urban environment equipped with high-resolution stereo cameras as well as five laser scanners [2]. Again, this work only employs a single camera from the stereo setup. Although there is no ground truth position available, very accurate GPS measurements are provided and can be used as a comparison to the position estimates.

C. Report structure

This report is structured in four sections: Section II provides an overview of the proposed VO and VIO pipeline. Afterward, a more detailed presentation of the algorithms used for each component is given. In Section III, encountered problems during the pipeline development are investigated and solution strategies are introduced. Followed by Section IV, where the results are presented and discussed. Finally, Section V provides some final remarks, outlines unresolved issues of our approach, and presents some possible future work.

II. Pipeline design

A. Overview

The building blocks of the proposed VO pipeline are implemented in the following classes:

1) `DatasetLoader`: Loading of images sequences, ground truth poses and calibration matrices.
2) `FeatureExtractor` and `FeatureMatcher`: Keypoint / Descriptor extraction and matching of two sets of features.
3) `BootstrapInitializer`: 8-point algorithm with RANSAC between 2 keyframes to extract an initial set of landmarks and the transformation between keyframes

4) `PoseEstimation`: PnP and KLT tracking.
5) `BundleAdjustment`: Local optimization of poses and landmarks.
6) `ContinuousVO`: Integration of the building blocks described above into a VO pipeline.

The interaction between the different elements is implemented in the `ContinuousVO` module, Fig. 8 presents a simplified overview of this process.

In general, images are processed sequentially, whereas for each frame keypoints are extracted and matched with the current landmarks. By having associated landmarks with key points in previous frames, continuous tracking over multiple frames is possible. Using the `PoseEstimation` module, keypoint correspondences are employed to estimate the transformation to the current frame. As the baseline uncertainty exceeds a certain threshold, the landmarks are updated by a re-bootstrap step. This step accounts for lost features during tracking and ensures consistent results. Local optimization is performed in a sliding-window manner by the `BundleAdjustment` module that refines both the landmarks position as well as the pose estimates in the window. Keyframe selection in order to initialize a set of 3D landmarks is performed manually and adapted empirically for the given dataset. The Bundle Adjustment step aims to tackle the scale drift and is part of the additional features implemented in this work.

Visual Inertial Odometry can be accomplished by a variety of methods, this work employs a Multi-State Constraint Kalman Filter [3], in particular, the stereo vision implementation by Sun et al. [4]. Using the work as Sun and its python implementation [5] as a baseline, the code has been adapted to work with a monocular setting and the presented datasets. This approach is part of the Extended Kalman Filter (EKF) methods. Hence, only one update iteration is performed and a fast execution time is ensured. Due to the additional information, the implementation differs from the VO pipeline by processing IMU measurements and image frames in parallel. More precisely, IMU measurements are handled immediately as they become available, for propagating the EKF state and covariance. On contrary, whenever an image frame is recorded, the current camera pose estimate is appended to the state vector and used to update multiple past poses in addition to the current state. The main building blocks are:

1) `ImageProcessor`: Handle features (recognize, match, prune) of the different frames and process the IMU data
2) `MSCKF`: state prediction and filter updates

B. Feature Extraction and Matching

As described above, the first part of the VO pipeline is to extract meaningful features from the input images. In this project we selected the Scale Invariant Feature Transform (SIFT) proposed by D.Lowe in [6] due to its robustness against 2D rotations, rescaling, illumination and moderate viewpoint changes. Figure 1 illustrates the SIFT descriptors detected in a KITTI frame.

Regarding the matching/tracking of features, two different scenarios are identified and treated with a different approach: *bootsrapping* and estimation with PnP.

- *Bootstrapping*: At the initialization phase or when the pipeline detects that a *rebootstraping* (see Section III-A)is needed, it is necessary to match the features of the keyframes to then perform the 8-point algorithm. Our approach uses the OpenCV library [7] feature matcher, `FlannBasedMatcher` which performs matching in a K-Nearest Neighbors fashion. In addition, a *kd-tree* is used to implement the K-Nearest Neighbors to boost the efficiency of the matching process. It is worth mentioning that our approach also implements a ratio test to only accept those matches whose $d_1/d_2 < TH$. The module `FeatureMatcher` encapsulates the mentioned approach.
- PnP estimation: When the pipeline is estimating the poses using the PnP algorithm and the previously computed set of landmarks our approach uses a KLT tracker to find the features of the past frame in the current frame. This process substitutes the normal matching algorithm and it is included in the `PoseEstimation` module. A detailed description of our KLT tracker approach can be found in Section II-D.

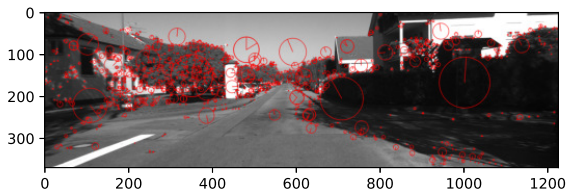


Fig. 1: SIFT features extracted from a single frame in the KITTI dataset.

C. Bootstrapping

The *bootstrapping* block has been implemented in such a way that it outputs the set of 3D landmarks and transformation between a pair of keyframes making use of the 8-point algorithm, given both keyframes and their calibration matrix. Thus, it can be treated as a black box by the other modules of the pipeline and can be separately tested and visually verified.

This module serves two purposes: the initial set of landmarks extraction and the *re-bootrstrapping* once the VO pipeline detects a certain decrease in the quality of the current landmarks with respect to the last processed frame. Regarding the initial bootstrapping, we followed the guide of the statement and skipped the first three frames of the dataset, which gives us a good enough baseline to compute the initial landmarks. As for the *re-bootstrapping*, Section III-A proposes a detailed discussion on the approach we followed.

The module workflow is summarized by the following steps:

1) Extract SIFT features and descriptors using the blocks described in Section II-B.
2) Compute the fundamental matrix using the self coded 8-point algorithm (resembles the *Exercise 6: Two-view Geometry* 8-point algorithm implementation but in Python).
3) Disambiguate the 4 solutions and extract the rotation matrix and translation vector.

The performance of this module can be tested by applying the algorithm to two different images with a correct baseline and visualizing its results. Figure 2 shows the performance of the algorithm with the KITTI dataset.

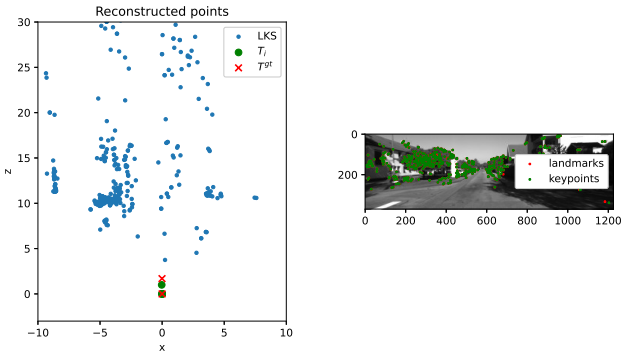


Fig. 2: Initial landmarks, poses and landmarks vs. keypoints of the second keyframe achieved using the *bootstrapping* block. Images taken from the KITTI dataset.

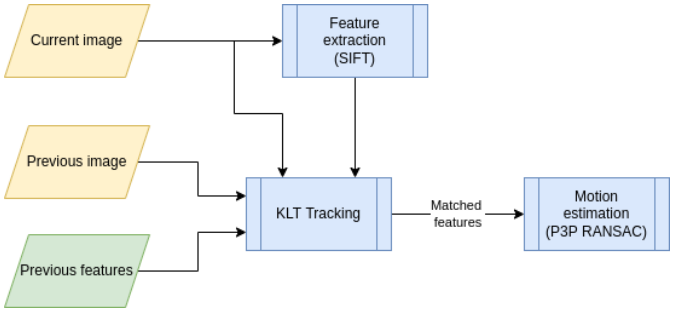


Fig. 3: Simplified diagram of the motion estimation process given two adjacent image frames.

D. Motion Estimation

The motion estimation module is the heart of the continuous operation phase, altogether with the *re-bootstrapping* explained in Section II-C. The module working principle is divided in two different jobs: the KLT tracking [8] and the PnP [9] algorithms. Figure 3 presents the workflow of this module.

When processing a new frame of the continuous operation phase, the KLT tracking is applied to find in the current frame the features of the previous frame which correspond to landmarks of the 3D landmarks, thus avoiding the feature matching process. In order to boost the efficiency of our pipeline, we decided to use the KLT implementation from the OpenCV library `calcOpticalFlowPyrLK`.

Once the KLT tracker has found the corresponding features in the current frame, the transformation between frames can be calculated using the PnP algorithm. This algorithm takes the current set of landmarks and the tracked keypoints found in the current frame and outputs the transformation between frames. Similarly to the KLT case, we use the OpenCV implementation of PnP with RANSAC to discard the outliers `solvePnPRansac` if the re-projected points distance is below a threshold $\delta_{max} \in \mathbb{R}$.

E. Bundle Adjustment

At this point the continuous VO pipeline could be executed with decently consistent results. However, a local optimization block was included to boost its performance and increment its accuracy. We propose a sliding window Bundle Adjustment optimization over a window of past poses and the landmarks

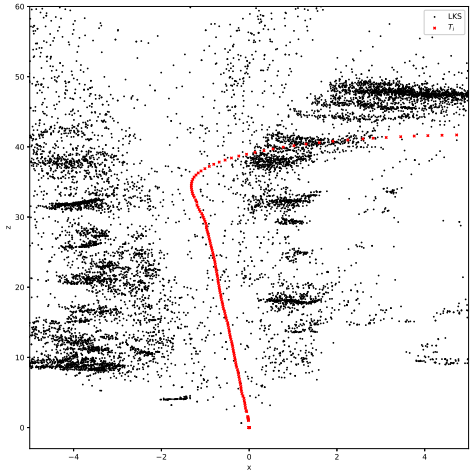


Fig. 4: Trajectory and landmarks for the first 150 frames of the KITTI dataset with Bundle Adjustment.

used for those poses to refine both the poses and the landmarks.

We took the Bundle Adjustment implementation of Scipy [10] and modified it in order to satisfy our needs, this implementation is quite similar to *Exercise 9: Bundle Adjustment*. We first apply a preprocessing to eliminate large errors that can make the optimization fail, thus we do not optimize over the reprojections that cause these large errors. In addition, we do not optimize over the first poses in order to avoid considerable discontinuities in the trajectory. This is easily achieved by setting the corresponding parts of the Jacobian to 0.

The `BundleAdjustment` module is called before every *re-bootstrapping*, thus we only optimize over a changing window of poses. This window consists on optimizing all the poses and landmarks compressed in the last $L \in \mathbb{N}$ keyframes. Therefore, after a local optimization a *re-bootstrapping* starts using the current frame and the previous keyframe as the new keyframes for the 8-point algorithm. This sliding window technique ensures that the optimization can be run online, as the optimization does not grow large enough so as to considerably slow down the pipeline.

Figures 4, 5, 6 and 7 show the performance of the VO pipeline with and without the sliding window bundle adjustment over the first few frames of the *KITTI* and *Parking* datasets, respectively. It is worth noting that the pipeline without the local optimization suffers some discontinuities, which is corrected by the `BundleAdjustment` block.

F. Continuous VO

As mentioned in Section II-A, the `ContinuousVO` module manages the continuous operation of the pipeline by using the above introduced modules. Figure 8 shows a simplified version of this control flow.

In more detail, after initializing the 2D-3D point correspondences by performing a bootstrapping step as discussed in Section II-C, it decides when to perform another round of bootstrapping, a step we call *re-bootstrapping*. We do so, whenever the the baseline σ_b between the last keyframe and the currently reconstructed pose (see Section II-D) is above

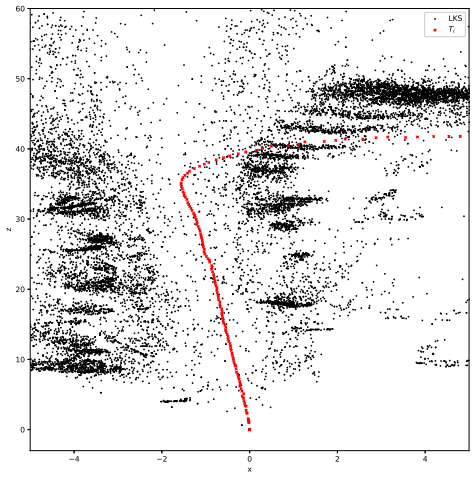


Fig. 5: Trajectory and landmarks for the first 150 frames of the KITTI dataset without Bundle Adjustment.

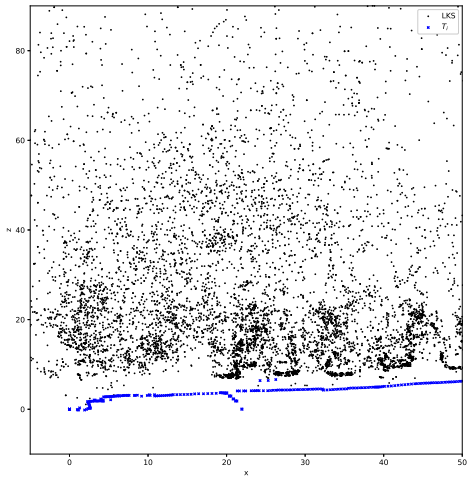


Fig. 6: Trajectory and landmarks for the first 230 frames of the Parking dataset with Bundle Adjustment.

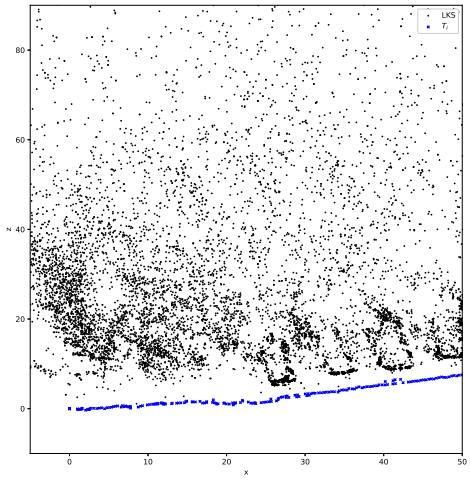


Fig. 7: Trajectory and landmarks for the first 230 frames of the parking dataset without Bundle Adjustment.

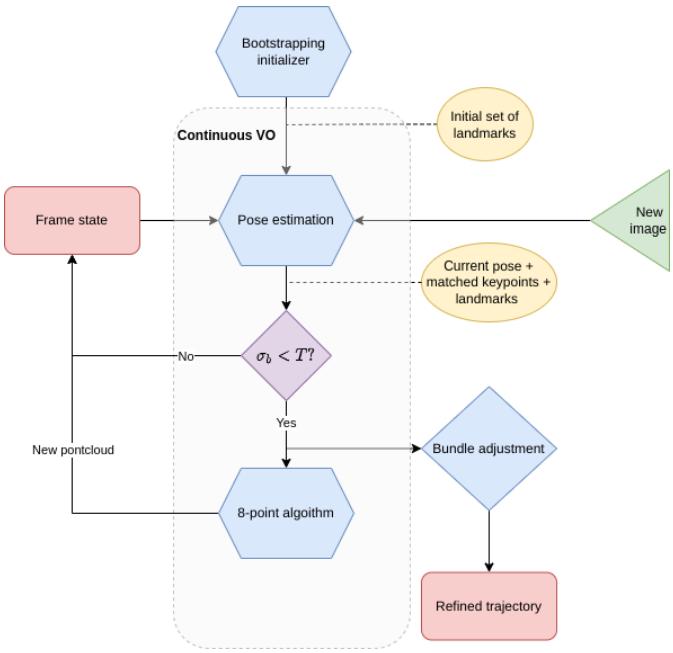


Fig. 8: Simplified diagram showing the continuous VO pipeline. Here σ_b denotes the baseline uncertainty and T a heuristic hand-tuned threshold to recompute the pointcloud.

a threshold T. At this point we call the `BootstrapInitializer` for *re-bootstrapping* and obtain a new set of 2D-3D correspondences from it. These are then tracked to succeeding frames, while keeping track of the new baseline.

When the baseline $\sigma_b > T$ and a *re-boostrapping* step is triggered, we additionally perform a local optimization step over all poses from the previous keyframe to the current frame, optimizing the re-projection of landmarks and pose estimates between each *re-bootstrapping*.

Besides managing these pose estimation steps, this module also keeps a record of the previously tracked poses in a limited horizon fashion as well as information on previous keyframes.

G. Visual Inertial Odometry

On top of the pipeline discussed in the previous sections, we also developed an independent extension of it, integrating IMU measurements to yield more accurate pose- and global scale estimates. In doing so we built on the work of Mourikis et. al. and Sun et. al. [3], [4], with the latter providing `C++` source code to their implementation. A `python` version from a different author is also available [5].

However, since the approach by Sun et. al.[4] requires stereo image sequences as well as IMU measurements, but the project statement explicitly prevents us from using a second camera stream, we removed all stereo routines in [5] and rewrote major parts of this implementation to yield a monocular VIO pipeline in python. To be able to compare trajectories of this distinct pipeline to ones by the VO pipeline discussed in the remainder of this report, we combined multiple versions of the *KITTI* dataset to obtain a version with the same rectified stereo images, but high frequency raw IMU measurements ($\approx$ 10 times image frequency). We call this dataset *KITTI IMU*. The resulting pipeline then, in essence works as the one introduced by Mourikis et. al [3] with the explicit integration of a simultaneous feature tracking and IMU state estimation using multiple threads.

However, likely due to incorrect transformations between the various IMU and camera reference frames, we were unable to correctly triangulate 3D landmarks, leading to poor pose estimation performance and do not evaluate its trajectories in detail. See more in Section III-C.

III. Encountered Problems & Solutions

A considerable amount of interesting problems arose during the implementation of each building block of the pipeline (see Section II), as well as when putting together all the blocks. This section presents a discussion on the problems we encountered and how we mitigate them to achieve the desired results.

A. Re-Bootstrapping

One of the biggest challenges we faced during the development of this pipeline was the the triangulation of new landmarks, as proposed by the project statement. We encountered major consistency problems of the triangulated landmarks when using the suggested keyframe trajectory approach.

These problems led to accumulating errors in the estimated trajectory, giving very bad pose accuracy when compared to the ground truth (e.g. on the *KITTI* dataset). We assume these inconsistencies came from small pose errors during tracking getting amplified by triangulating new landmarks only with respect to the first and last observed pose for any given trajectory. Since such a newly triangulated point would then be used for succeeding pose estimates, these errors accumulated very quickly.

Therefore, we changed our approach to implement the continuous operation phase by *bootstrapping* and tracking the resulting 2D-3D correspondences with KLT, then estimating the resulting pose via PnP. However, instead of adding new features over time we *re-bootstrap* and replace all landmarks on each new bootstrapping phase, which led to a considerable increase in the robustness of our pipeline.

To decide when to *re-bootstrap* we adopt the concept of baseline uncertainty σ_b introduced in the lecture. This method computes the L_2 distance d between the current frame and the previous keyframe as well as the average landmark depth with respect to the last keyframe.

$$\sigma_b = \frac{d}{\frac{1}{N}\sum_{i=1}^{N} P_z^i} \tag{1}$$

Where P_z^i is the z component of the i-th landmark in the reference frame of the previous keyframe (assuming the z-axis points forward). of these values. In addition we define a minimal ratio I_{min} of landmarks that need to be tracked from the previous bootstrapping phase to the current frame. If ratio of tracked points is less than I_{min} or σ_b is greater that a threshold T, a *re-bootstrapping* phase is performed.

B. Scale Drift

As any monocular VO pipeline, our pipeline is prone to scale drift over time. Since our basic pipeline was very prone to this issue we tried two different strategies to mitigate this issue: (i) local optimization of the pose estimates between keyframes (Bundle Adjustment) and (ii) Visual Inertial Odometry, i.e. integrating IMU measurements to get high frequency estimates of the global scale.

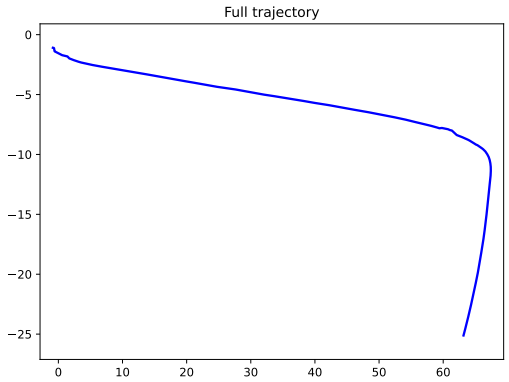


Fig. 9: VIO trajectory on the first 20s of the *KITTI IMU* dataset.

The first approach consists of implementing the bundle adjustment module detailed in Section II-E. By improving the pose estimates between keyframes jointly when minimizing the re-projection error δ, Bundle Adjustment greatly improves the local consistency of our resulting pose estimates as can be seen in Figure 6. However, it does not help to reduce scale drift, which we could only lower by decreasing the baseline uncertainty threshold T, therefore running the *bootstrapping* step more often.

C. VIO integration

As mentioned in Section II-G we additionally tried to integrate IMU measurements with the pose estimation steps by coupling them together tightly. However, when integrating the *KITTI IMU* dataset with the approach by Sun et. al. [4], we had issues updating the estimated IMU state from the tracked image features. This was likely the case, as the reconstructed 3D landmarks were not projected to the correct world coordinate system as seen in Figure 10. Here, the predicted trajectory is orthogonal to the reconstructed landmarks, even though in the image sequence the vehicle is driving straight. These pose errors become even more apparent, once we consider a longer trajectory in Figure 9.

The cause of these errors is most likely due to drift exhibited by the IMU, which is not corrected by the feature tracking pipeline, since the tracked feature velocities do not align with the expected IMU velocities. This causes all such samples to be rejected in the approach by Sun et. al [4]. We could confirm this behavior by observing the very low frequency of updates from feature tracking to IMU state estimates.

Therefore, the results we see in Figure 9 and Figure 10 mimic the ones which would be produced by an loosely coupled VIO pipeline, leading to very large pose estimation errors.

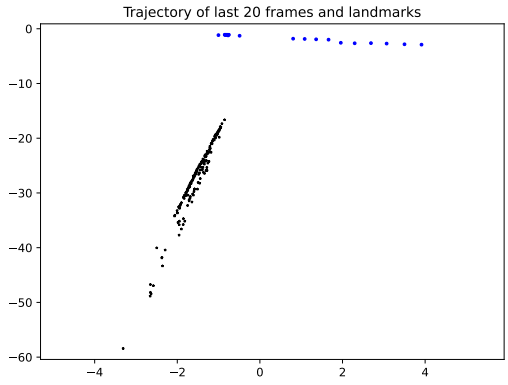
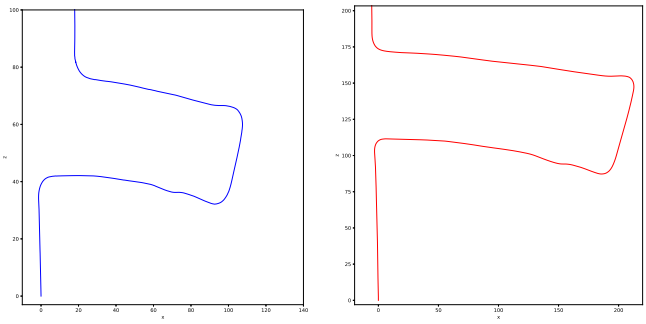


Fig. 10: VIO trajectory in *KITTI IMU* dataset in blue and reconstructed 3D landmarks in black. Landmarks are reconstructed orthogonal to direction of travel (right).



(a) VO estimated trajectory (b) Ground Truth

Fig. 11: VO and ground truth trajectory of the first 850 frames of the KITTI dataset.

IV. Results

In this section the results achieved by our VO pipeline in the used datasets are presented. The discussion of the results relies on a qualitative analysis of the performance achieved by our pipeline.

In general, the proposed pipeline achieves a locally consistent pose estimation performance in both datasets, even though scale drift is present. The mentioned behaviour is shown in Figures 12, 13 and 14, which present the results of the pipeline with the KITTI, Málaga and Parking datasets, respectively. In addition, Figure 11 provides the estimated trajectory of the VO pipeline in comparison with the ground truth trajectory of some frames of the KITTI dataset. These figures demonstrate that the VO pipeline achieves the desired performance at any point of the trajectory. Clearly, the performance of the VO pipeline diminishes when the number of tracked keypoints in a frame considerably decreases. However, our approach tackles this problem with the *rebootstrapping*, which generates a new set of landmarks and, in its turn, increases the number of tracked keypoints. Therefore, the overall performance is still consistent.

Note that our pipeline is robust to all motions present in the datasets, such as tight turns or continuous straights. Furthermore, the Málaga dataset contains large illumination changes over time which are difficult to handle for pixel intensity based methods, however our proposed pipeline is robust to these kind of difficulties as well. In addition, the number of detected keypoints with the *bootstrapping* phases considerably decreases in some frames of this same dataset. When this could be a potential setback for VO pipelines, our approach still estimates the poses in a consistent manner.

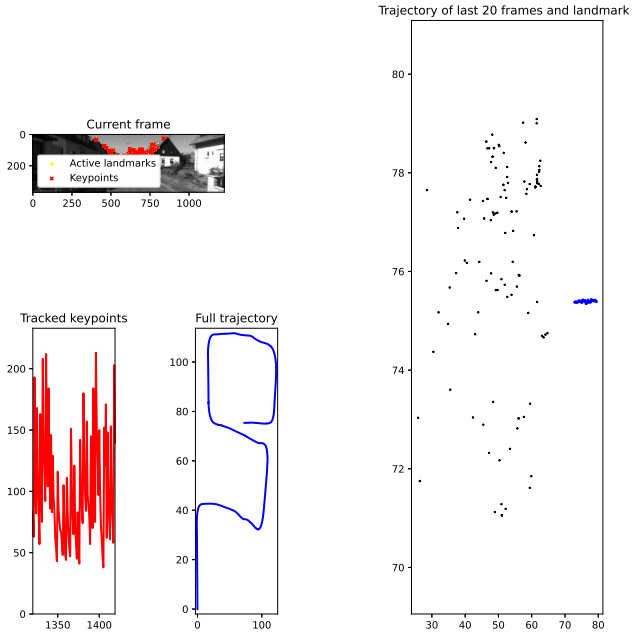


Fig. 12: VO pipeline output using the KITTI dataset.

Dataset Name	σ_b	I_{min}	L	δ_{max}
KITTI	0.15	0.5	4	1
Parking	0.15	0.5	20	1
Málaga	0.15	0.5	4	1

TABLE I: VO Pipeline parameters used to generate the trajectory. σ_b: Baseline Distance Threshold, I_{min}: Minimal Inlier ratio, L: Bundle adjustment look back, δ_{max}: RANSAC re-projection threshold.

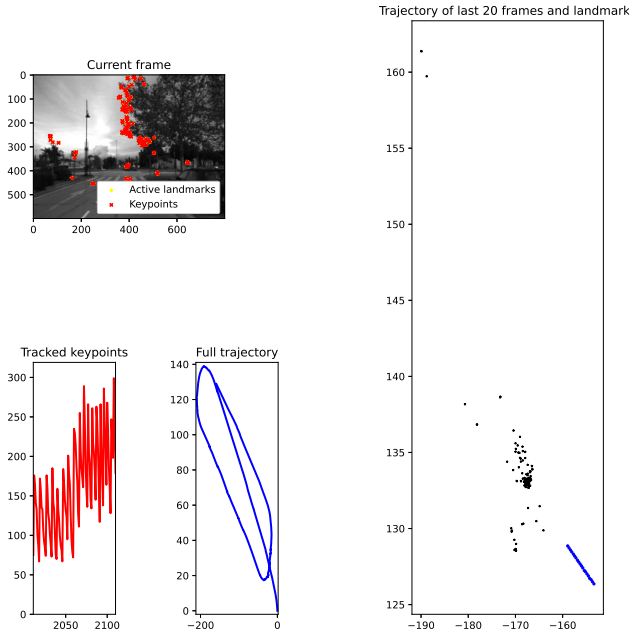


Fig. 13: VO pipeline output using the Málaga dataset.

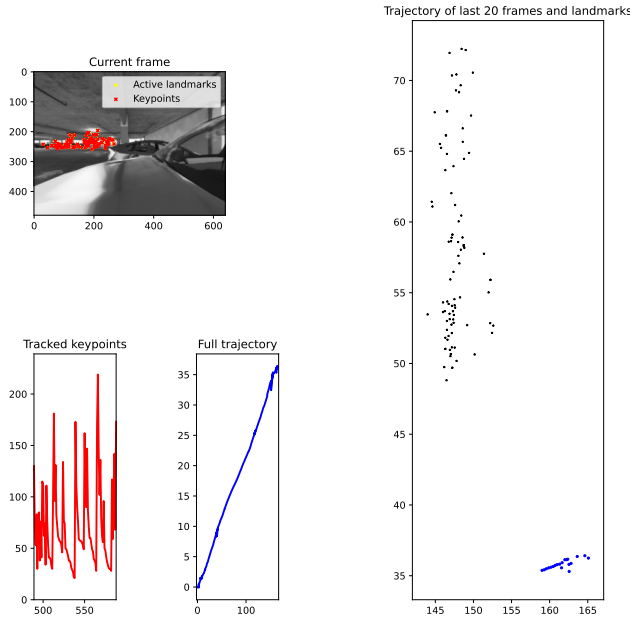


Fig. 14: VO pipeline output using the Parking dataset.

V. Conclusion and Future Work

Overall, we showed during this report that the proposed VO pipeline achieves local consistency of the estimated trajectory on all three datasets accomplishing the main goal of the project. Taking robustness as the main priority of the pipeline, well-established techniques that have proven their reliability achieving a consistent performance across the three datasets as shown in Section IV. We want to especially highlight the importance of the customized Bundle Adjustment (BA) approach, which greatly helped to reduce the accumulated error of integrating raw VO outputs. Also notable is our exploration of a tightly coupled IMU pipeline based on the MSCKF [3]. Despite not improving the local consistency of the system, the sensor fusion approach notably allowed us to estimate the global world scale and increased the robustness by introducing a fallback system in the event of vision failures. Therefore, given the constrained time-frame of the project, we consider the proposed goals as fulfilled and the project having surpassed it's initial scope.

For future work, we consider the definitive removal of scale drift. As an idea, the implementation of the closed form solution presented by Martinelli in [11] would give the scale parameter needed to correct the motion estimates. Secondly, we would like to improve the performance of the VIO implementation to make sure that the local consistency of the pipeline is also improved by this fusion, as well as looking into global pose optimization methods.

References

[1] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[2] J.-L. Blanco, F.-A. Moreno, and J. Gonzalez-Jimenez, "The málaga urban dataset: High-rate stereo and lidars in a realistic urban scenario," *International Journal of Robotics Research*, vol. 33, no. 2, pp. 207–214, 2014. [Online]. Available: http://www.mrpt.org/MalagaUrbanDataset

[3] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, pp. 3565–3572, ISSN: 1050-4729. [Online]. Available: http://ieeexplore.ieee.org/document/4209642/

[4] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, "Robust stereo visual inertial odometry for fast autonomous flight." [Online]. Available: http://arxiv.org/abs/1712.00036

[5] "uoip/stereo_msckf: Python implementation of multi-state constraint kalman filter (MSCKF) for vision-aided inertial navigation." [Online]. Available: https://github.com/uoip/stereo_msckf

[6] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94

[7] OpenCV, "Open source computer vision library," 2015.

[8] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision (ijcai)," vol. 81, 04 1981.

[9] R. Fischler and M. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun ACM*, vol. 24, pp. 619–638, 01 1981.

[10] N. Mayorov. Large-scale bundle adjustment in scipy. [Online]. Available: https://scipy-cookbook.readthedocs.io/items/bundle_adjustment.html

[11] A. Martinelli, "Closed-form solution to cooperative visual-inertial structure from motion," 2018.