University of Zurich[UZH]

A Monocular Visual Odometry Pipeline

by

Hardik Shah

Deepak Ganesh

Aniruddha Sundararajan

Deepana Ishtaweera

A report submitted as part of
the mini-project for the course

Vision Algorithms for Mobile Robotics (Fall 2023)

Department of Informatics

January 2024

Contents

Abstract

Localization through odometry state estimation is a key and fundamental component in autonomous systems. Monocular ***Visual odometry*** *(VO) is the task of estimating the 6DoF motion of a camera using only a single image sequence. In this mini-project, we present a pipeline for Monocular VO that consists of an initialization component and a continuous pipeline. The former initializes the landmarks and the camera poses for the continuous pipeline to track the key points and landmarks and keep triangulating new landmarks while estimating the correct camera pose. We complement our implementation with some additional features such as testing it on our own dataset captured using a Realsense D435i camera, Re-Initialization, etc. Monocular VO is a challenging problem due to the scale ambiguity, susceptibility of outliers in feature matching, and dynamic scenes with varying lighting conditions. Moreover, it also suffers from scale drift where the errors in camera pose estimates accumulate over time and cause the scale of the reconstructed scene to change inconsistently. There are different methods to address the scale drift issue such as using additional sensors like IMU or LIDARs, or Deep Learning, etc. But, under the constraint of monocular vision, we try to combat this issue by attempting to use Bundle Adjustment as another feature. We showcase the results of our pipeline on the KITTI, Malaga and Parking datasets.*

1 Problem Statement

In the rapidly advancing field of mobile robotics, the implementation of efficient and reliable navigation systems remains a paramount challenge. This mini-project focuses on developing a foundational component of such systems: a monocular visual odometry (VO) pipeline. Visual Odometry plays a crucial role in the perception and navigation of autonomous robots, particularly in environments where GPS is unreliable or unavailable.

The core objective of this project is to create a simple yet effective VO pipeline, incorporating the essential elements of monocular visual odometry. These elements include:

- Initialization of 3D Landmarks (Bootstrapping): Establishing initial reference points in the environment that the camera can recognize and track across different frames.
- Keypoint Tracking Between Two Frames: Detecting and matching salient features or keypoints across consecutive image frames to track the movement and rotation of the camera.
- Pose Estimation Using Established 2D $\leftrightarrow$ 3D Correspondences: Determining the camera's position and orientation by aligning 2D key points from the image frame with their corresponding 3D points in the environment.
- Triangulation of New Landmarks: Continuously updating and expanding the map of the environment by triangulating new landmarks as the camera explores.

We integrate these components into a cohesive streamlined system, building upon the fundamental concepts and techniques explored in exercises of the Vision Algorithms for Mobile Robotics course.

This report summarizes the details of our work and describes the additional features that we developed. We implement our solution in an Anaconda environment with Python 3.9. In addition to the features recommended in the project statement, we employ re-initialization to reduce losing track of visual odometry due to low number of landmarks, and bundle adjustment (BA) to refine the trajectory. Furthermore, we evaluate our approach on a custom dataset that encompasses a 400m path, constituting a rectangular loop with open spaces and ample lighting. The dataset is acquired by navigating around the residence of a team member using a handheld Realsense D435i camera. The screencasts can be viewed at this link, and the code is made available at this github repository.

This report is structured as follows: Initially, we discuss the various components of the VO pipeline and our approach to implementing them. This includes a detailed explanation of the specific functions employed and the overall design of the pipeline. Subsequently, the report delves into the fine-tuning of parameters, addressing both the initialization phase and continuous operation aspects of the project. The report concludes with an evaluation of the screencast results and an analysis of the performance using the benchmark datasets and our custom dataset.

2 Implementation

This section contains details regarding the implementation of the proposed visual odometry pipeline. The pipeline consists of two major components, **VO Initialization** which bootstraps the initial camera poses and 3D landmarks using 2D↔2D correspondences, and sets the scale for the subsequent poses (section 2.1). Next, **Continuous Operation** which estimates the pose of the camera at each frame, and triangulates new landmarks (section 2.2). The high-level workflow of the pipeline is depicted in figure 2.1.

2.1 VO Initialization

For establishing the 2D↔2D correspondences, sufficiently distant two initial frames are selected by manual selection. The first frame (I_0) is the first image of the dataset (left image if the dataset is stereo). The second frame (I_1) is selected by observing the images and selecting the image with sufficient motion with respect to the first frame(which ensures a sufficiently large baseline for initial point triangulation). The choice of second frame was found to be an important factor that affects the accuracy of the trajectory. This is discussed in detail in Chapter 3. Next, keypoints (image coordinates and descriptors) are extracted from the two images using a feature detector. The pipeline was tested with different detectors such as SIFT, Harris corners, and Shi-Tomasi corners. As bootstrapping is done only at the initial stage, SIFT features were promising despite the higher computational time considering robustness to viewpoint and illumination changes, and invariance to scale and rotation changes. To match features, bruteforce matching (`cv2.BFMatcher().knnMatch()`) is used by taking the two nearest matches in the second image for a keypoint detected in first image and using a distance threshold factor of 0.8 to extract strong matches. Figure 2.2 shows keypoint matches overlayed on the second frame (I_1). The unmatched keypoints in I_1 are initialised as the set of candidate keypoints C, which denote potential keypoints that can be triangulated in the future frames.

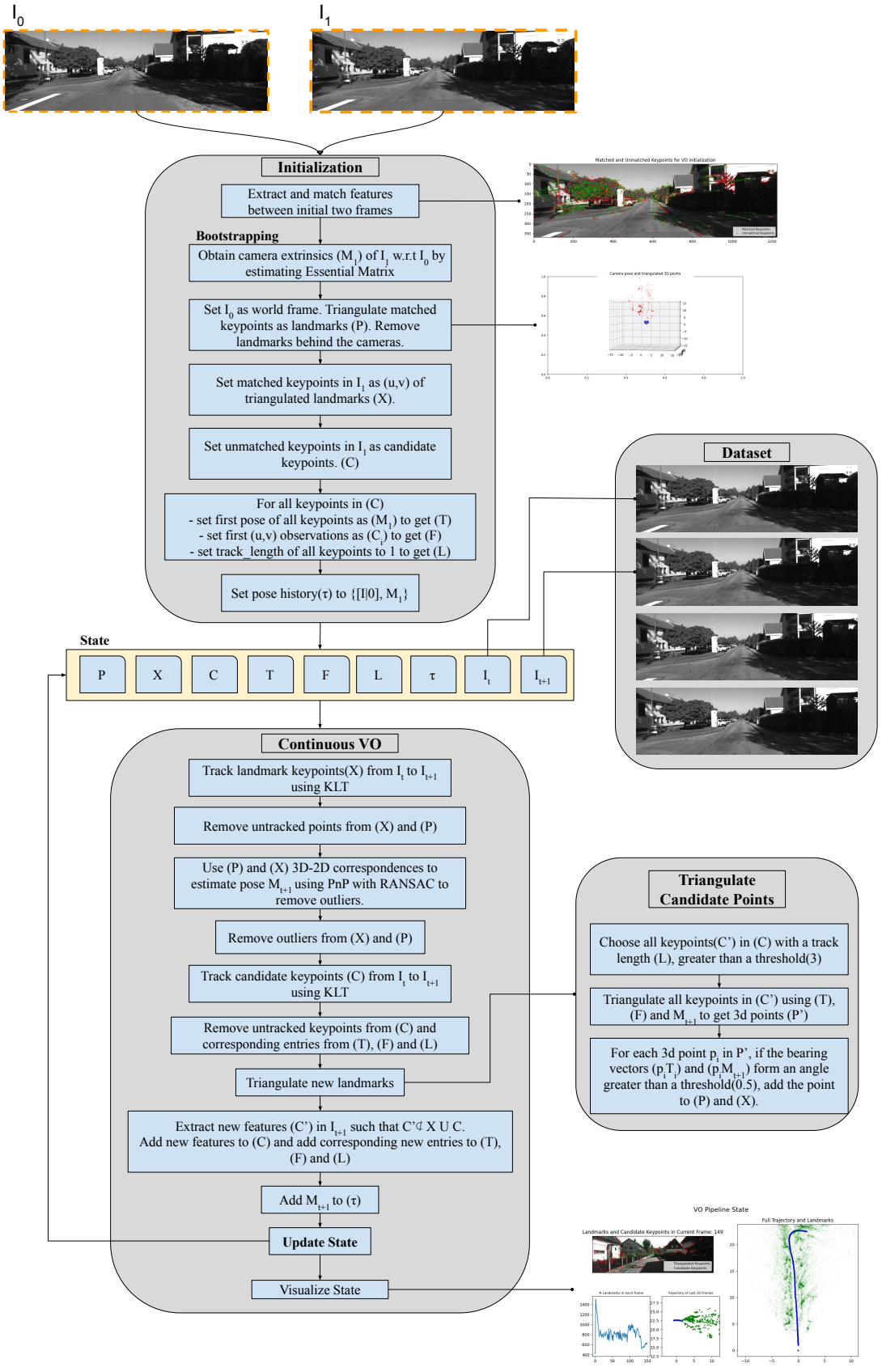


FIGURE 2.1: Workflow of the VO pipeline

We initialise the pose of the first frame as $[I|0]$ i.e. the world frame, and the essential matrix (`cv2.findEssentialMat()`) is estimated using the 5-point algorithm which gives us the pose of I_2. We use RANSAC to filter out the outliers in keypoint matching. Using the essential matrix, the pose of the second camera is retrieved (`cv2.recoverPose()`) considering the first frame as the world frame. Finally, we triangulate the matched keypoints (after removing the outliers given by RANSAC) using `cv2.triangulatePoints()` to obtain the 3D landmarks. We discard landmarks triangulated behind the cameras. This set of 3D points is used to initialise the set of landmarks P, and their corresponding keypoint locations as X. Figure 2.3 shows the triangulated 3D points and the camera poses for I_0 and I_1 of the KITTI dataset.

We initialize the state of the pipeline at frame t as illustrated in figure 2.1:

$$S = [P, X, C, T, F, L, \tau] \tag{2.1}$$

1. P: **3D landmarks** tracked in the current frame
2. X: **(u,v) locations** of the keypoints in the current frame corresponding to P^i
3. C: (u,v) locations of the set of **candidate** keypoints in the current frame
4. T: **camera pose of the first frame** for each keypoint in C where it was first detected
5. F: (u,v) locations of the **candidate keypoints in the first frame**, corresponding to entries in C
6. L: **track length** of each keypoint in C
7. τ: the **history of poses** of the previous frames i.e. the trajectory till t

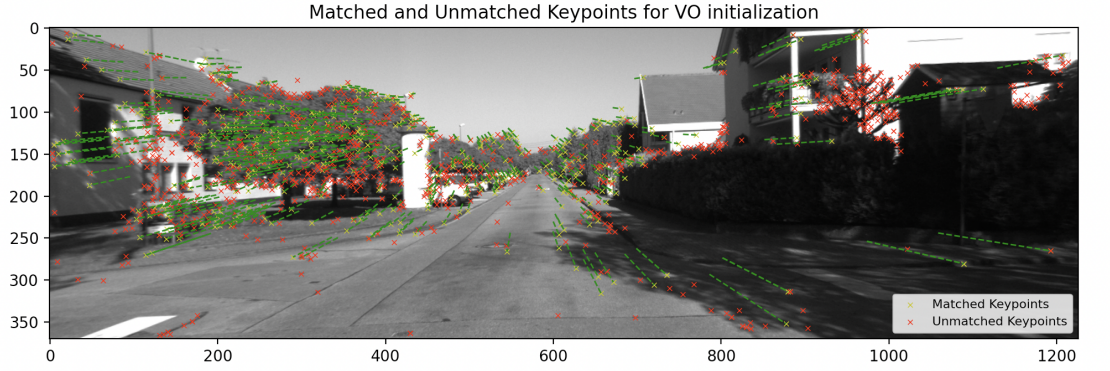


FIGURE 2.2: Keypoint matches for VO initialization: matched keypoints are marked in yellow 'x', unmatched keypoints are marked in red 'x'. Green track lines connect the keypoints in the first frame to the second frame.

2.2 Continuous Operation of VO

The continuous VO block of the pipeline is the focal component of our visual odometry (VO) implementation, performing three key functions:

1. It associate keypoints in the current frame with pre-existing triangulated landmarks, establishing the 2D↔3D correspondences.
2. Utilizing these alignments, it computes the camera's current position and orientation.
3. It systematically triangulates new landmarks using keypoints that are not associated with previously triangulated landmarks.

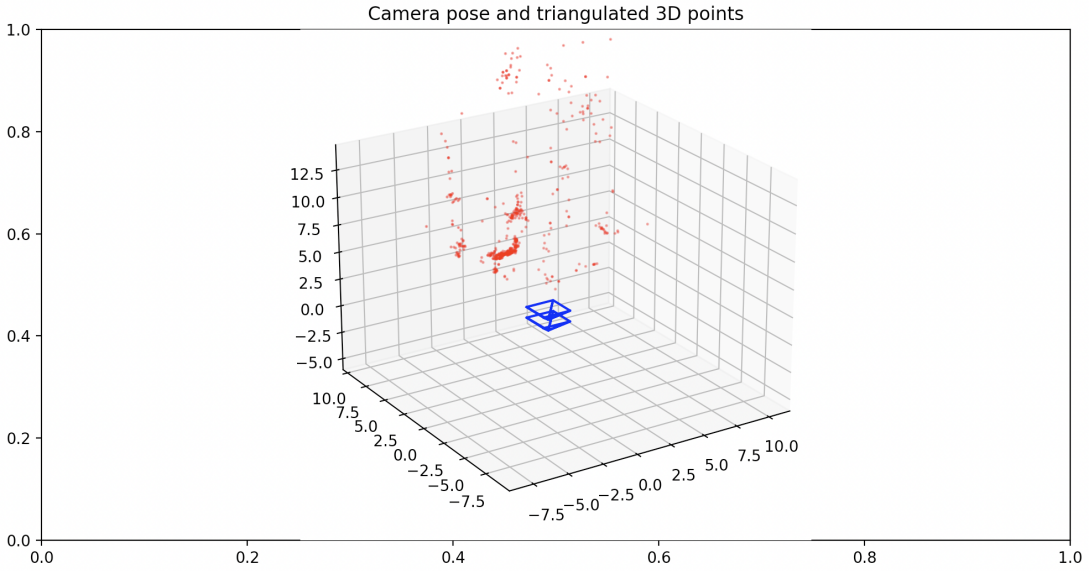


FIGURE 2.3: Initial set of 3D landmarks: oriented upward to view the two initial camera poses easily. The first camera frame is the world frame (I_0) and the second camera frame(I_1) is in front of it which shows the motion in the forward direction. The triangulated landmarks are shown in red dots.

The implementation is designed to exhibit the Markov property (2.2), where given the current image frame I_t, previous image frame I_{t-1}, previous state S_{t-1}, we return the current state S_t and the current global camera pose T_t^{WC}. The description of the state S_t is given in equation 2.1.

$$[S_t, T_t^{WC}] = \text{processFrame}(I_t, I_{t-1}, S_{t-1}) \tag{2.2}$$

We track the landmark keypoints (X_{t-1} of the previous frame(I_{t-1} using KLT (`cv2.calcOpticalFlowPyrLK()`) to the current frame I_t. We remove the untracked landmarks from the sets P and X. Using the 2D↔3D correspondences (P and X), the camera pose (T_t^{WC}) is estimated using the PnP algorithm (`cv2.solvePnPRansac()`) with RANSAC. The outlier landmarks given by RANSAC are removed from P and X. Next, the candidate keypoints C_{t-1} from the previous frame are tracked to the current frame similarly. The unmatched candidate keypoints are dropped from C, T, F and L. Keypoints that are tracked for a minimum track length are triangulated. This triangulation step is done using the C, F, T and T_t^{WC}. The landmarks that have a bearing angle ($\alpha(c)$) (defined in figure 2.4) higher than a threshold, and are triangulated in front of the cameras are selected as the new landmarks. P and X are updated to include these new landmarks.

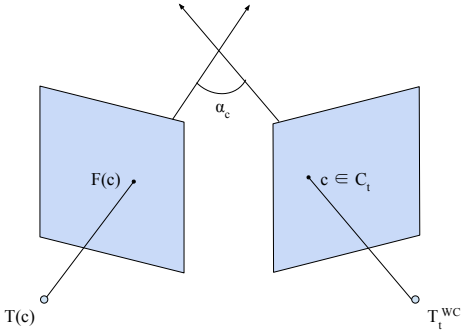


FIGURE 2.4: llustration of the bearing angle α_c which we threshold on for deciding whether or not to add a triangulated landmark and its keypoint to X, P

To generate new candidate keypoints, we run feature detection on the current image frame. This is tested with SIFT, Harris corners, and Shi-Tomasi corners. The selection of the type of features depends on the dataset which is discussed in the chapter 3. To ensure that the new candidate keypoints are unique and do not coincide with the set if current landmarks and candidates, we must choose new candidate keypoints C' such that $C' \notin X \cup C$. For the same, we pass a mask which masks the (u,v) locations of keypoints in C and X to the feature extractors. The new candidate keypoints are added to C, and the corresponding values also added to T, F and L. The visualization of the continuous operation of the VO pipeline is depicted in the figure 2.5.

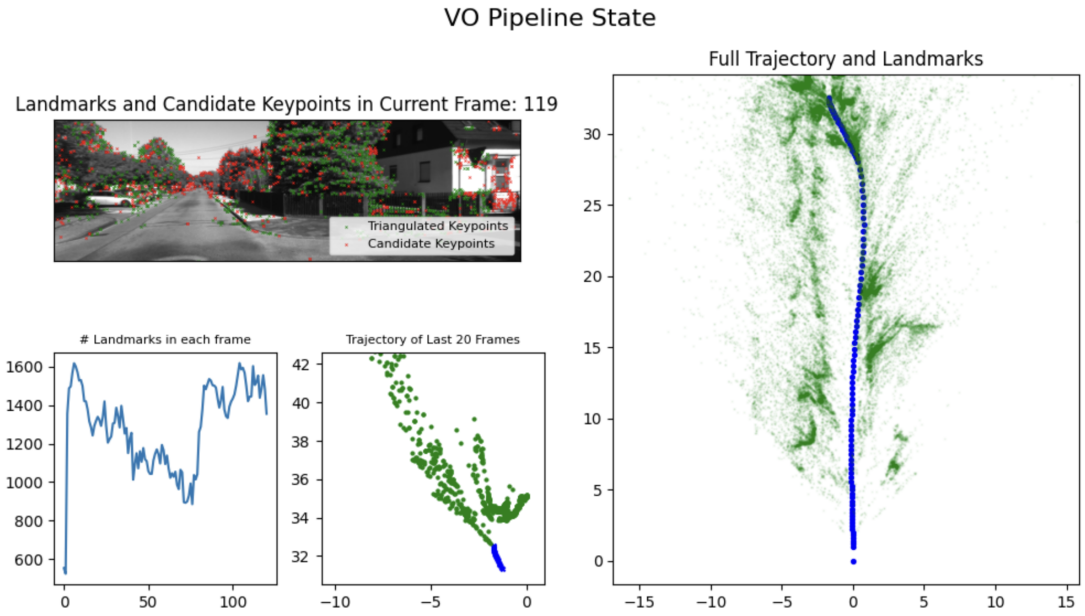


FIGURE 2.5: Visualisation of the continuous operation mode on KITTI: top left shows the landmarks and candidate keypoints in the current frame (similar to figure 2.2, bottom left shows the line chart of number of landmarks present at each frame, bottom middle shows the local trajectory of the last 20 frames with landmarks, right shows the global trajectory with landmarks.

3 Parameter Tuning and Evaluation Results

To achieve the optimal performance for each dataset in our pipeline, we fine-tuned various hyperparameters targeting specific challenges within each dataset. Specifically, we experimented with various angle thresholds, choice of initial frames for initialisation, window size in the Kanade–Lucas–Tomasi feature tracker, the maximum number of features returned by feature extractor(`nFeatures`).

Additionally, we applied a novel trick to reduce the number of outlier features being added to the set of landmarks and lower the cost of triangulating them. The `min_track_length` hyperparameter sets the minimum number of consecutive frames across which a feature must be tracked before it gets added to the set of landmarks. This ensures that the landmarks predominantly come from an inlier feature.

We also tried out different feature extractors such as SIFT, Harris and Shi-Tomasi corner detectors. Shi-tomasi and Harris feature extractors gave us significant speedup in the runtime compared to SIFT. However, the quality of the features extracted by SIFT were better, which led to better tracking with KLT. Based on the dataset, we selected the extractor that provided an optimal balance in the tradeoff.

One of the main challenges faced by our model was the lack of enough landmarks. The model either lost track of saved landmarks or could not convert candidates into landmarks. We noticed that this usually happened in sections of the data set with low lighting conditions or glare. Increasing the maximum number of features returned by feature extractor mitigated this issue, but it did not solve it completely.

Initially, we also faced a drop in landmarks when the vehicle makes a turn. We fixed this issue by removing the bidirectional error check in the KLT tracker and increasing the maximum number of features returned by feature extractor.

3.1 Parking dataset

The Parking dataset presents an easy baseline to test our model. A primary challenge presented by this dataset is that the features and thus landmarks in the initial set of frames are situated at a considerable distance from the camera. We tackled this issue by setting a longer initial baseline, which lead to a lower drift. We employ a SIFT feature extractor for this dataset. Figure 3.1 shows the final trajectory obtained by the model with the surrounding landmarks highlighted in green.

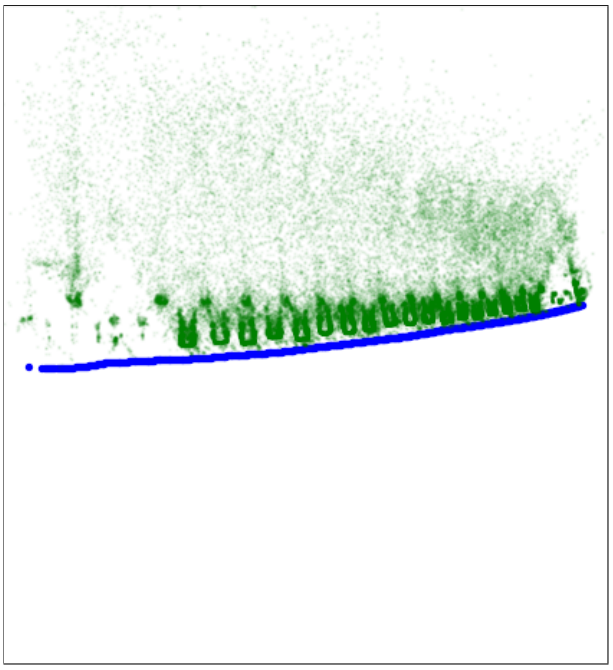


FIGURE 3.1: Final Trajectory for the Parking dataset with landmarks

3.2 Kitti dataset

The Kitti dataset presents a hard baseline to test our model. Many scenes in this dataset capture moving cars and objects, which led to scale drift. Furthermore, whenever the camera encounters a straight road with numerous trees on the sides, the model faces a significant amount of outliers. Consequently, this results in the failure of the KLT tracker, causing a sudden drop in the number of landmarks. In order to solve this issue, we re-initialize the landmarks when the number of landmarks drop below a certain threshold. We discuss this feature in detail in section 4.1 of Chapter 4. We employ a SIFT feature extractor for this dataset. Figure 3.2 shows the final trajectory obtained by the model with the surrounding landmarks highlighted in green.

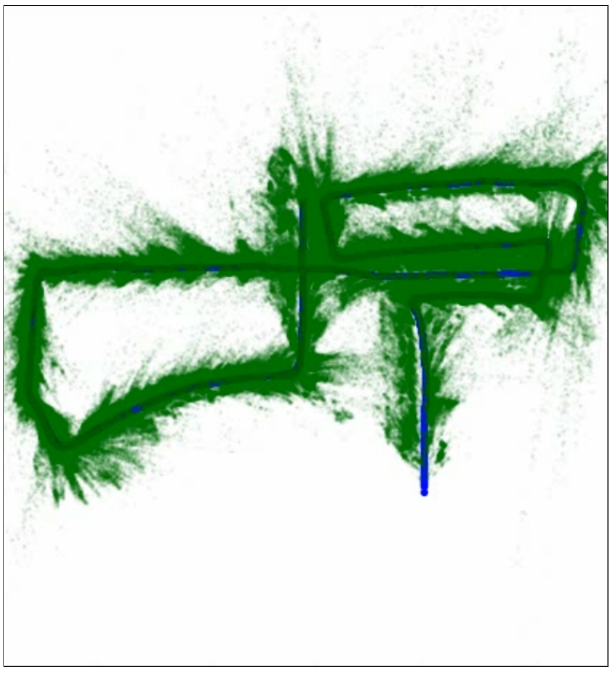


FIGURE 3.2: Final Trajectory for the Kitti dataset with landmarks

3.3 Malaga dataset

The Malaga dataset presents another hard baseline to test our model. We employ a Shi-tomasi feature extractor for this dataset. Figure 3.3 shows the final trajectory obtained by the model with the surrounding landmarks highlighted in green.

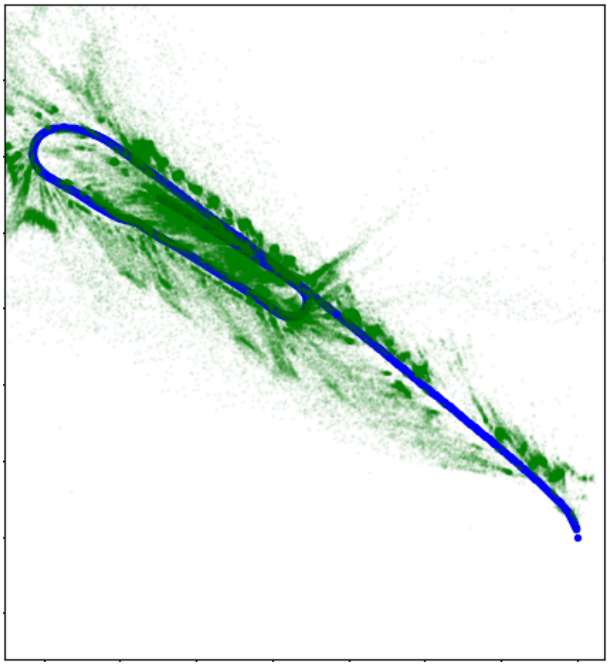


FIGURE 3.3: Final Trajectory for the Malaga dataset with landmarks

4 Additional Features

Observing the defects and issues (chapter 3) in the implementation of the VO pipeline described in figure 2.1, below are some techniques we used to tackle scale drift, and sudden drop in landmarks and candidates. We also discuss the evaluation of our pipeline on a custom dataset.

4.1 Re-initialisation

One major challenge we faced during evaluation of the pipeline on the KITTI dataset, was the sudden drop in landmarks, sometimes directly to 0, which ended up in either the pipeline failing or an erroneous pose estimation from PnP (figure 4.1). We fixed this issue temporarily by tuning parameters like number of features in the feature extractor, window size of KLT etc. as described earlier in chapter 3. This drop can be attributed to major part of the image being textureless regions, and sudden change of illumination in the scene.

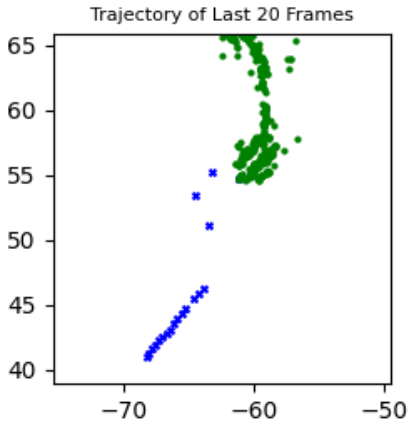


FIGURE 4.1: Erroneous Pose estimate in the KITTI dataset after sudden drop in landmarks.

We propose to circumvent this situation by re-initialisation of the landmarks when the number of landmarks drop below a certain threshold. The re-intialisation is quite similar to the VO intialisation discussed earlier in 2.1, in the sense that we again establish 2D↔2D correspondences between the current frame where we lost track of landmarks and a previous frame (need not be subsequent, we keep that parameter tunable). The key challenge here is to compute the correct scale. Since, the pose estimate we recover from the essential matrix is **up to a scale**, and we have already defined a scale in the VO intialization step (we set the translation between I_0 and I_1 as a unit vector). One could calculate $T_{C_2C_1}$ where C_1 denotes the previous frame we use for re-initialization and C_2 denotes the current frame, and calculate $T_{C_2W} = T_{C_2C_1} x T_{C_1W}$ where T_{C_1W} can be used from the pose history we maintain in the VO pipeline. But this is would be a false estimate of the pose of the current frame.

We implemented two ideas for re-intialisation, both of which work decently in some scenarios and prevent the pipeline from failing due to the sudden drop in landmarks:

1. **Compute Relative Scale between 2D↔2D and 2D↔3D estimates**: triangulate new landmarks and refine current pose estimate using the essential matrix estimate and scaling it by the computed scale factor
2. **Triangulate new landmarks** Initialize new 3D landmarks by brute force matching, but do not refine the current pose

We observed method 1 to work better than method 2. However, the global trajectory is disturbed after the re-initialisation step, but since this method is to stop the pipeline from a complete failure it works well.

For computing the relative scale factor for method 1, we use the distance ratio (r) between pairs of points in the 3d points we have from P of the pipeline and the pairs of newly triangulated points in re-initialization, as discussed in [1]. Assume, we are re-initializing using the current frame I_t and a previous frame I_{t-k} for a positive integer k. We must first find common 3D landmarks between these two frames, and hence we update the state representation to keep track of the history of 3D landmarks. Let the set P_1 denote common 3D landmarks between I_t and I_{t-k}. Let x_t and x_{t-k} denote the corresponding (u,v) locations of the landmark keypoints in the two frames. We use x_t and x_{t-k} as 2D↔2D correspondences to estimate the essential matrix, and initialise a new set of 3D points(upto a scale) P_2 using the new pose estimate. Now, for each pair of points in P_1, we calculate the distances between these points and construct set D_1. Similarly we calculate D_2 for P_2. Now, we take the mean(or median) of the ratio of corresponding distances between D_1 and D_2. This represents the scale factor(r) for the new relative pose estimate between I_t and I_{t-k}. We scale the translation part of this pose estimate with r and multiply it with T_{t-k}^{CW}. Now, to triangulate new landmarks we do brute force matching for features in both these two frames, and triangulate using the new refined pose for I_t, and update the state.

4.2 Bundle Adjustment

Bundle adjustment (BA) is a technique that optimizes multiple camera poses and the 3D landmarks simultaneously by minimizing the reprojection error between the 2D points projected from the landmarks and the keypoints that correspond to them in a given frame. It is often used in visual odometry (VO) to reduce the drift error that accumulates over time due to frame-by-frame pose estimation. Scale drift is an unavoidable problem in VO and Bundle Adjustment is one way of dealing with it.

We initialize the state of the pipeline at frame t as illustrated in figure 2.1.

$$S = [P, X, C, T, F, L, \tau, \zeta] \tag{4.1}$$

Additionally, we define the extra state ζ to maintain a history of landmarks for the past n_{camera} frames to prepare the data for performing Bundle Adjustment. The subset of the (tracked inlier + newly triangulated landmarks) that are outliers in the next frame are then saved in the history of landmarks in the current frame. In this way we make sure we don't have any duplicates of the landmarks, and also because (tracked inlier + newly triangulated landmarks) in the next frame are considered as the valid landmarks in the next frame, but the outliers aren't.

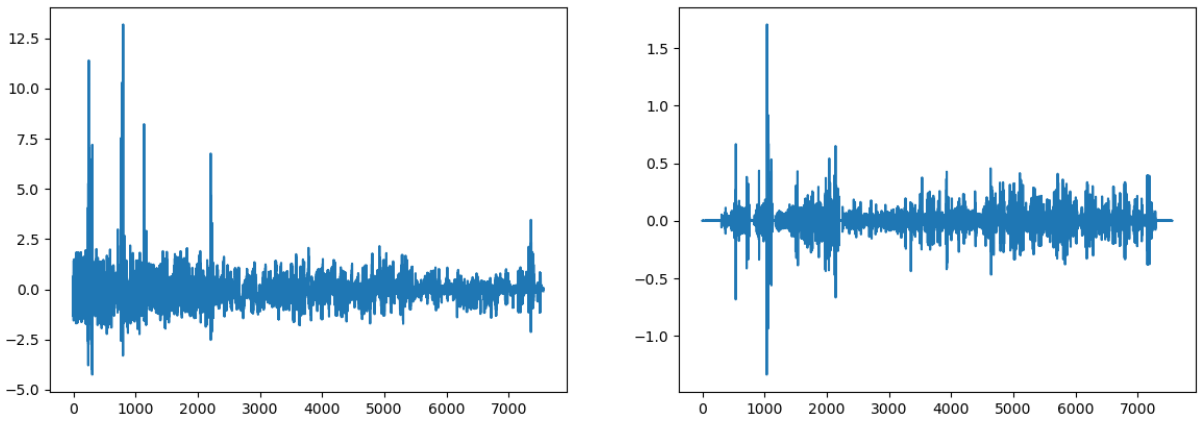
1. ζ: the **History of Unique Landmarks** of the previous frames i.e. the trajectory till t
2. n_{camera}: the **Number of frames** that we collectively **bundle adjust** over.

Trust Region Reflective (TRF) algorithm is a method of solving nonlinear least squares problems with bounds on the variables. Bundle Adjustment is a nonlinear least squares problem with bounds on the

variables, such as the camera focal length, rotation angles, and translation vectors. The usage of sparse Jacobians can speed up the convergence of the TRF algorithm because it can efficiently handle large-scale problems with many parameters and constraints. For our pipeline, we gain inspiration from [2]'s implementation of Bundle Adjustment, by using (`scipy.optimize.least_squares`) with (`'trf'`) as a flag.

Our implementation of Bundle Adjustment, although decent enough, suffers majorly from the high computational time. Eventhough it was run on a laptop with a decent CPU, Bundle Adjustment takes 5 seconds for optimizing over camera poses and landmarks of the past 10 frames. Since, the computation time is an issue, we couldn't utilize Bundle Adjustment to it's fullest, and it doesn't prove to be an upgrade over our current working pipeline. With a Multi-threading approach (mentioned in section 5.1 as future work) and a faster implementation we can have a much powerful Monocular VO pipeline.

The initial reprojection errors are not too large, which supports the performance of our current pipeline. In Figure (4.2) we can verify the correctness of the implementation since the reprojection errors always get smaller after adjusting. We can still observe a few outliers from the result. This could be because of the algorithm getting stuck at a local minima or the convergence being slow. The algorithm used an estimate of the Jacobian, which might not be very accurate near the solution and slow down the algorithm since there doesn't exist a closed form solution to the Jacobian in the algorithm.



(A) Before Bundle Adjustment

(B) After Bundle Adjustment

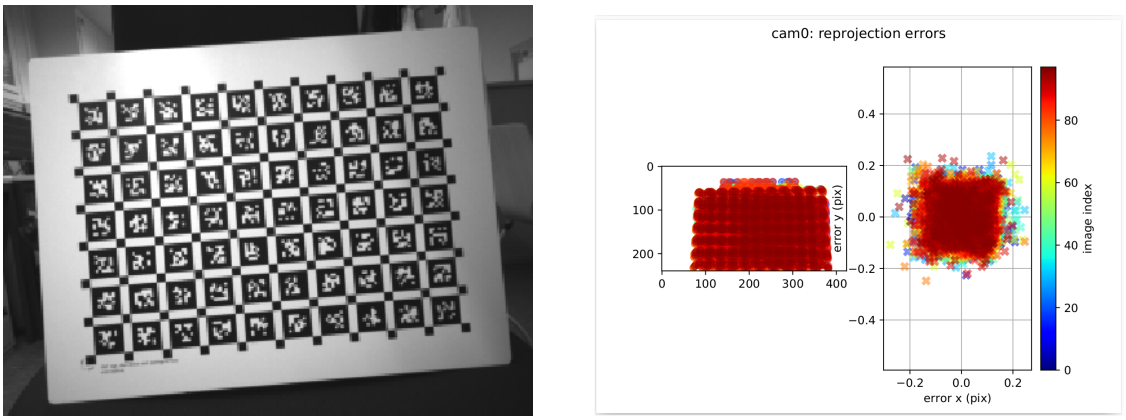
FIGURE 4.2: Plot of Reprojection error vs Total number of residuals

4.3 Custom Dataset

The custom dataset, essential for testing our visual odometry (VO) pipeline, was meticulously captured in the vicinity of the residence of a team member, characterized by open space and ample lighting. The dataset encapsulates a 400-meter path, forming a rectangular loop. The recording was accomplished using a Realsense D435i camera which equipped with an IMU and a stereo camera. This camera was mounted in front of a bicycle while being connected to a latop during the recording. For our purposes, we utilized only the images from the left camera, received at a frequency of 15Hz. This dataset is comprehensive, comprising approximately 2600 images. Furthermore, the recording was saved as a ROS bag using the realsense sdk, and was later converted into a directory of images.

A critical aspect of our dataset preparation involved camera calibration. This was conducted using an April Tag fiducial marker board. We recorded a separate rosbag specifically for this calibration process. The calibration software used was "kalibr" [3], a well-regarded ros package available on GitHub for camera calibration. This software facilitated the extraction of precise camera intrinsics, which were pivotal for

the accurate functioning of VO pipeline. Figure 4.3 shows the camera reprojection error results after successful calibration. Find the snippet of the video recorded for the Calibration here.



(A) April Tag Board

(B) Reprojection error after calibration

FIGURE 4.3: Calibration of the Realsense D435i camera using [3]: Sub one-pixel reprojection error shows the accuracy of calibration

This dataset resembles kitti and malaga in the sense that it was real, and kitti more because it was recorded around a residential area with trees and grasses, and plain houses. Trees and Grasses are known for their textureless surfaces and this always causes an issue with feature detection and tracking because of it's ambiguity in patch matching. Moreover, the dataset is also recorded at a resolution of (640×480) which is not the most idea for performing Visual Odometry. The stark observation in this dataset was the significant scale drift with the scale decreasing throughout the recording as seen from figure (4.4). This was because of poor features impacting the results to produce poorer landmarks during triangulation. An ideal Odometry pipeline should have closed the loop when returning to the starting point, but our Visual Odometry pipeline with it's own drawbacks, unfortunately suffered from scale drift and couldn't achieve it.

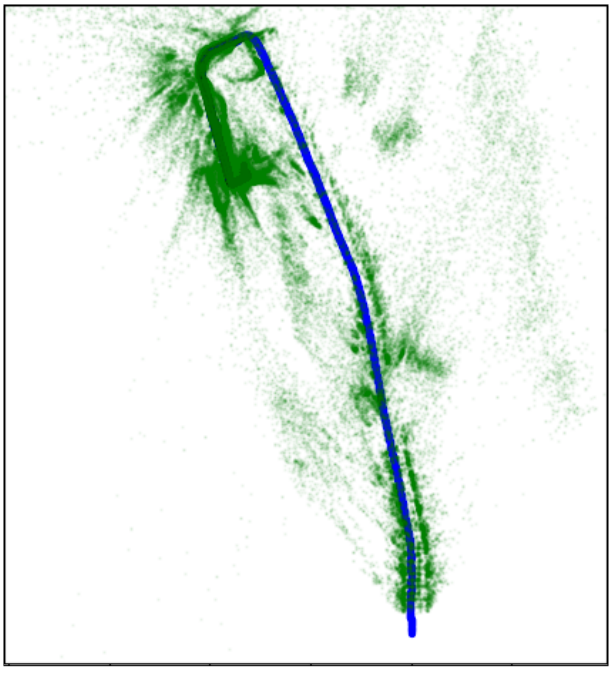


FIGURE 4.4: Final Trajectory for the Custom dataset with landmarks

5 Conclusion

We saw that our monocular visual odometry pipeline is able to achieve locally consistent trajectories. We encountered various challenges in developing a functional version of the pipeline, and this report detailed the solutions we have put forward to overcome these obstacles. Moreover, this project has laid a strong fundamental understanding of the concepts taught in the course Vision Algorithms for Mobile Robotics.

5.1 Future Work

The current implementation of Monocular VO in this project has potential for a lot of improvements. Through the course of this project, several ideas could not be implemented due to time restrictions. A lot of concepts taught in the lecture can be applied to the current implementation in order to tackle issues like scale drift, and assessment of the trajectory returned by VO.

1. **Multithreading in Bundle Adjustment:** Running bundle adjustment on a separate thread will speed up the pipeline, enhancing real-time performance.
2. **Non-linear Refinement in Triangulation:** Replacing the least squares estimate with a non-linear approach that minimizes the reprojection error can increase the accuracy of landmark positioning, thereby reducing errors in triangulation and improving the trajectory.
3. **Removing Scale Ambiguity:** Datasets like KITTI and Malaga, give access to the absolute scale by giving information about the placement of the cameras on the car. This information can be used during initialisation and also during continuous VO to remove scale ambiguity and preventing scale drift. This would include detecting the ground plane in the triangulated keypoints, or specifically detecting keypoints on the road to compute the scale factor.
4. **Leveraging Constrained motion of cars:** Since most datasets (except ones recorded using a handheld camera) are recorded using cars, the motion of the camera is constrained to translation in XZ plane, and if we don't consider flat ground then rotation around the X axis. These additional constraints could be leveraged for better pose estimates.
5. **Quantitative Assessment of VO and Stereo Vision Integration:** For some datasets like KITTI, access to the groundtruth trajectory is available. A quantitative assessment of the VO trajectory would be interesting. Another additional ablation study that could be performed is comparison of the monocular VO trajectory with a stereo VO trajectory. Stereo vision will provide depth information, improving the system's robustness in depth estimation addressing the scale drift.
6. **Incorporating Additional Techniques from the course:** Utilizing advanced methods from the course, like improved feature matching and motion models, can optimize the pipeline's efficiency.
7. **Loop Closure with Place Recognition:** Implementing loop closure will correct trajectory drift by recognizing revisited locations, thereby improving long-term accuracy.

We plan to maintain the Github repository as an open-source project and continue contributing to it to integrate the above mentioned action items.

5.2 Author Contribution

In this mini project, each team member played a pivotal role, contributing unique skills and expertise. The following table outlines the specific contributions of each member (the first letter of the first name and the last name is used in the column name):

Table 5.1: Author Contributions

Work Package	**HS**	**DG**	**AS**	**DI**
Importing and Extracting Data	-	✓	-	✓
Define Code Structure	-	-	✓	✓
Implement Bootstrapping	-	-	✓	-
Implement Pose Estimation and Triangulation	✓	-	-	-
Extraction of Keypoints with Feature Detectors	✓	✓	-	-
Pipeline Integration	✓	✓	✓	✓
Implement Bundle Adjustment	-	✓	-	-
Implement Re-initialisation	✓	-	-	-
Testing and Refining on KITTI, Malaga, and Parking Dataset	✓	✓	✓	✓
Setup YAML Configuration Files	-	-	✓	✓
Recording Custom Dataset and Camera Calibration	✓	✓	✓	✓
Testing and Refining on Custom Dataset	✓	✓	-	-

Bibliography

[1] Friedrich Fraundorfer Davide Scaramuzza. Visual odometry part i: The first 30 years and fundamentals. URL https://rpg.ifi.uzh.ch/docs/VO_Part_I_Scaramuzza.pdf.

[2] Large-scale bundle adjustment in scipy. URL https://scipy-cookbook.readthedocs.io/items/bundle_adjustment.html.

[3] Autonomous Systems Lab, ETHZ. Kalibr. URL https://github.com/ethz-asl/kalibr.