Locally Consistent Visual Odometry Pipeline: Project Report

Zeno Hamers, Riccardo Maggioni, Augusto Mondelli, Nicola Taddei

January 7, 2024

1 Introduction and project overview

Visual Odometry (VO) is a critical technique in robotics, autonomous navigation, and computer vision, enabling devices to determine their position and orientation in GPS-denied environments, solely based on camera movements. This document reports on the implementation of a simple monocular visual odometry pipeline, tested on the KITTI [2], MALAGA [1] and parking datasets. This report is written as part of the Vision Algorithms for Mobile Robotics class from Prof. D. Scaramuzza.

The VO pipeline has been modularly constructed, consisting of one initialization part and 3 continuous operation parts (constructed in a Markovian fashion). For each module, several alternative components were explored and parameter optimization was performed to optimise the performance of the pipeline. The final VO system shows local consistency on the KITTI, MALAGA and parking datasets.

This report consists of the following three sections:

Implementation Demonstrates the VO pipeline and explains algorithmic choices

Results and Discussion Showcase and discuss the VO results

Conclusion and Future Work Conclude on the project and recommend future work

Author Contributions

Here, the work packages and main responsibilities of the authors are listed:

Zeno Hamers Implemented the initialization Class, worked on the new candidate keypoints (continuous operation) implementation, wrote substantial amounts of the report, conducted parameter testing, merged modular parts into a working pipeline

Riccardo Maggioni Implemented the KeypointsToLandmarksAssociator Class, created plotting visualizations, merged modular parts into a working pipeline, worked on the report

Augusto Mondelli Implemented the LandmarkTriangulator Class, conducted parameter testing, worked on the report

Nicola Taddei Implemented the Pose Estimation (Continuous Operation) Class, Designed the System Architecture (initial setup)

2 Implementation

In this section, the implementation of the VO pipeline is demonstrated and the algorithmic choices are explained. The BestVision pipeline is encapsulated in the `BestVision` class, designed as a modular and extensible framework for visual odometry. The class consists of 4 parts: 1. Initialization, 2. Keypoint Associations (KLT), 3. Pose Estimation and 4. New Landmark Triangulation. Of these 4 classes, the last 3 are part of the continuous operation. The code implementation of this project can be found in the complementary repository of this report. Each of the following subsections will tackle one of the four modular parts of the project, starting with the initialization.

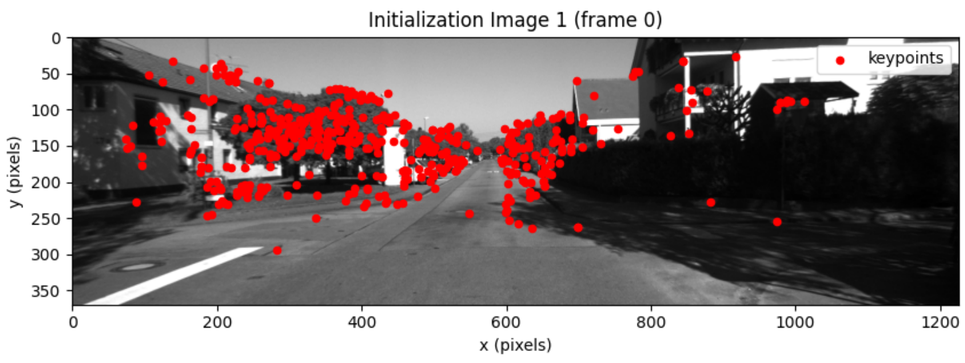
2.1 Initialization (bootstrapping)

During the initialization process (also called bootstrapping), the position of the second frame is estimated and a first 3D point cloud is generated. This 3D point cloud is then saved as a landmark 'X', used in the continuous operation to estimate the pose of the consecutive frames using the PnP algorithm. The most successful initialization frames were [0,6] for the KITTI dataset, [0,6] for MALAGA and [0,4] for the Parking dataset. These images were visually inspected to make sure enough frame-to-frame motion (i.e. baseline) was present. Since the continuous operation process relies on the KLT tracker, which is highly dependent on the initial value, an accurate initialization is paramount. To establish this precise initialization, we opted for an indirect approach (which can cope with larger frame-to-frame, providing a larger baseline for triangulation). This approach consisted of three main steps: 1. establishing keypoint correspondences, 2. estimating the pose of the second frame, 3. triangulating the point cloud of 3D landmarks

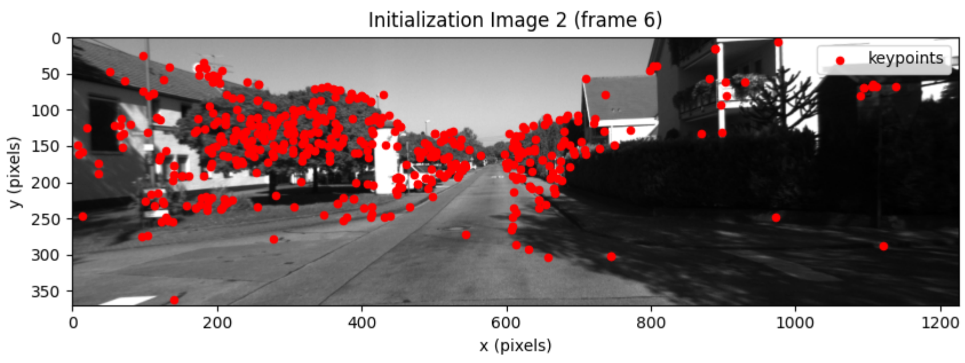
2.1.1 Establishing keypoint correspondences

For maximum robustness of the initialization, we opted for the SIFT blob detector (over the Shi-Tomasi corner detector) in combination with the complementary SIFT feature descriptor to establish 2D-2D correspondences. The SIFT descriptor is well known for its robustness against viewpoint and illumination changes. Keypoint correspondences were obtained by brute force matching of the SIFT descriptors, using the ratio test as the selection criteria for good keypoint matches. Both the SIFT blob and the Shi-Tomasi corner detector were tested in combination with the sift descriptor for the preciseness of the eventual pose estimation. The SIFT detector proved to give more 2D correspondences and accurate triangulated 3D landmarks. Figure 2 depicts the comparison between the Shi-Tomasi and the SIFT detector for initialization in the pipeline.

Figure 1 shows the keypoint correspondences of the two initialization frames [0,6] for the KITTI dataset with the SIFT detector (in combination with the SIFT descriptor). The keypoint correspondences for the Shi-Tomasi detector can be found in the appendix A.



(a) SIFT Keypoint correspondences in the first initialization frame (00) of the KITTI dataset.



(b) SIFT Keypoint correspondences in the second initialization frame (06) of the KITTI dataset.

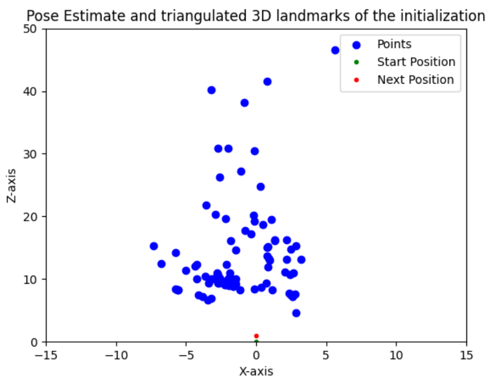
Figure 1: SIFT (detector and descriptor) keypoint correspondences of the two initialization frames [0,6] of the KITTI dataset.

2.1.2 Pose Estimation of the second frame

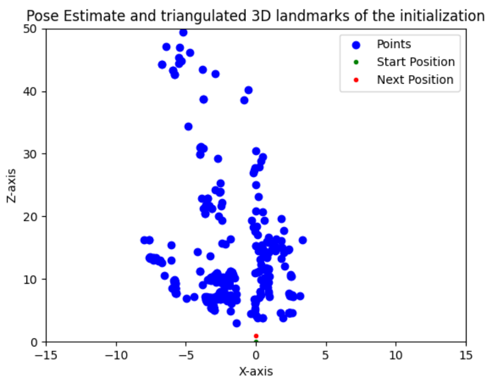
Since all three datasets provided the camera calibration matrix K, the 5-point algorithm was used to estimate the position of the second frame relative to the first one (exploiting the epipolar geometry). To exclude outliers, a RANSAC (confidence threshold of 0.999) was implemented.
From the calculated essential matrix, 4 possible combinations of R and T are possible, but only one is in front of both cameras. The pose of the second frame (R, t) was recovered by applying the cheirality condition (cv2.recoverPose).

2.1.3 Triangulation of 3D landmarks

The 3D landmarks of the initialization were constructed by triangulating the keypoint correspondences of the first 2 frames (position of the second frame estimated). Figure 2 shows the triangulated 3D landmarks of the initialization for the KITTI dataset, depicting both the SIFT and Shi-Tomasi detector implementation.



(a) Shi-Tomasi detector for initialization



(b) SIFT detector for initialization

Figure 2: Triangulated 3D landmarks (blue) and estimation pose (red) of the initialization frames (origin: frame 0, red dot: frame 6) for the Shi-Tomashi and the Sift Detector respectively (in combination with the SIFT descriptor). The SIFT detector was selected for the final pipeline.

2.2 Continous Operation: Keypoint Associations (KLT)

Following the Markovian design, keypoint association is critical for updating the state with validated keypoints. The `KeypointsToLandmarksAssociator` module tracks keypoints from the old image to features in the new image. Optical flow is employed to estimate the new positions of keypoints. The resulting associations are used to update the state.

2.3 Continuous Operation: Pose Estimation

Pose Estimation addresses the problem of recovering the camera pose associated with a new frame. 3D landmarks produced by other components of the pipeline and features corresponding to those points are used in a localization scheme. The EPnP algorithm is used alongside with 3 point ransac and a final nonlinear refinement is applied. This last step involves only the camera pose, as the position of the 3D landmarks is considered to be sufficiently accurate. All these functionalities are implemented by the OpenCV function `solvePnPRansac`. The behaviour of the function depends mainly on two tunable parameters:

- **reprojection error threshold**: from this parameter depends how many points are accepted as inliers. Ideally one wants to have the lowest value but, due to noise of various kind, we had to keep it at 2, otherwise we don't have enough inliers.
- **ransac confidence**: the confidence we espect the output of the ransac to have. A high number causes an increase in the number of iterations, but a low one gives unreliable solutions. The final pipeline used a value of 0.99999 for all the datasets.

2.4 Continuous Operation: New Landmark triangulation

The landmark triangulation module (`LandmarkTriangulator`) reconstructs 3D landmarks from candidate keypoints. It maintains a list of candidate keypoints and their transformations (T). The module tracks candidate keypoints over frames, validating and updating their positions. It also adds newly tracked candidates to the system. The process ensures the maintenance and expansion of the map of 3D landmarks.

We decided to implement the suggested approach based on creating a set of candidate keypoints that allowed to maintain a Markovian propriety of the pipeline. The set of candidate keypoints was initialized with new keypoints that were tracked in the frames that followed the two used to create the initial point-cloud. All candidate keypoints were associated with the camera pose matrix corresponding to their first observation and the pixel value of their first observation.

Then, at every iteration, all the 2d points in the candidate set were tracked from the previous to the next image by using KLT tracking, which allowed a fast tracking and an update of the pixel position of all candidate keypoints in the current frame, as well as the removal of all candidate keypoints that were not tracked again. In this step we used the opencv function *calcOpticalFlowPyrKLT()*.

At this point we proceeded to check which candidate points to validate: we computed the angle between the bearing vector corresponding to the current observation and the one corresponding to the current observation. We then confronted this value with a threshold value alpha (in degree) and proceeded to validate the 2d keypoints that managed to have an angle between the above-mentioned vectors greater than the threshold. If this check was successful, we proceeded to triangulate this points with opencv function *triangulate-Points()*. After a few tries, we decided to set the threshold angle alpha to a value of 1 degree, which allowed us to maintain a (rather high) number of validated 2d keypoints and 3d landmarks in the state, sufficient to allow us to always estimate the pose with PnP.

In order to add new keypoints to the candidate set, we decided to use SIFT to detect all the keypoints in the new frame and compute their descriptors, which were then saved as an attribute of the class object so that it was not necessary to recompute them at the following iteration. These descriptors were then matched by brute force matching to the stored descriptors of the previous frame, so that new candidates were taken by filtering all the keypoints in the current image that were not matched with keypoints in the old one.

After this process, we proceeded to add the detected new candidates to the candidate set by adding their pixel position in the current frame as the pixel position of the first observation, as well as the current camera pose matrix as the camera pose of the first observation.

With this choice of parameters we managed to get local consistency in the straights and the corners: figure 3 shows the first three turns in the KITTI dataset.

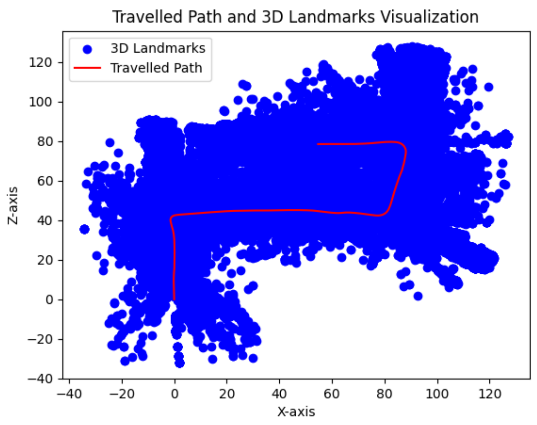
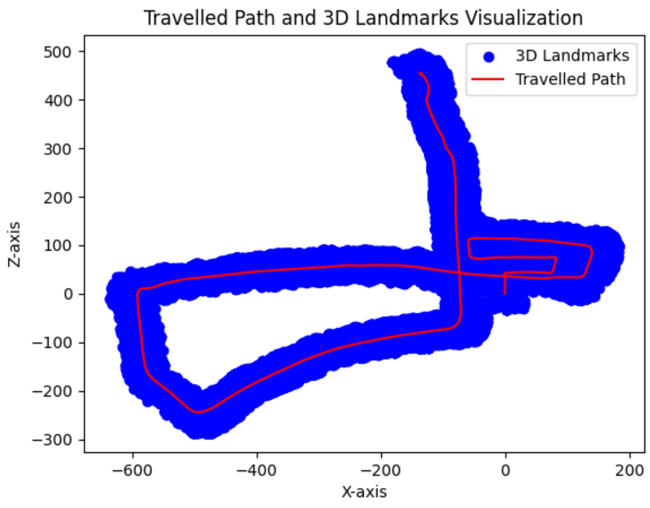


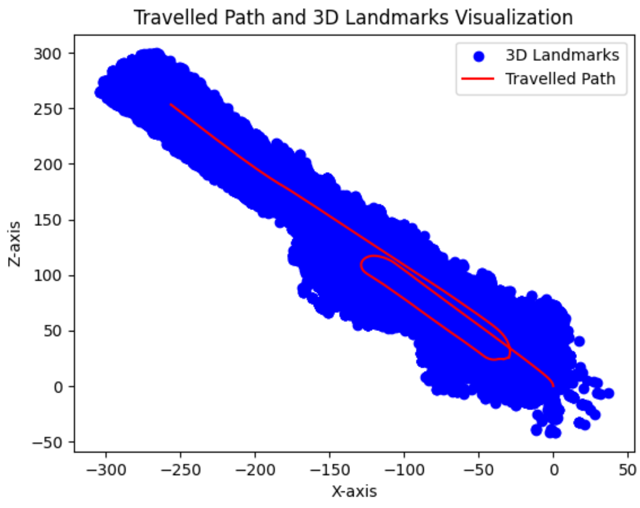
Figure 3: Screencast showcasing local and global trajectory, together with the keypoints and 3D landmarks, for the first three turns in the KITTI dataset. Clear local consistency is shown.

3 Results and Discussion

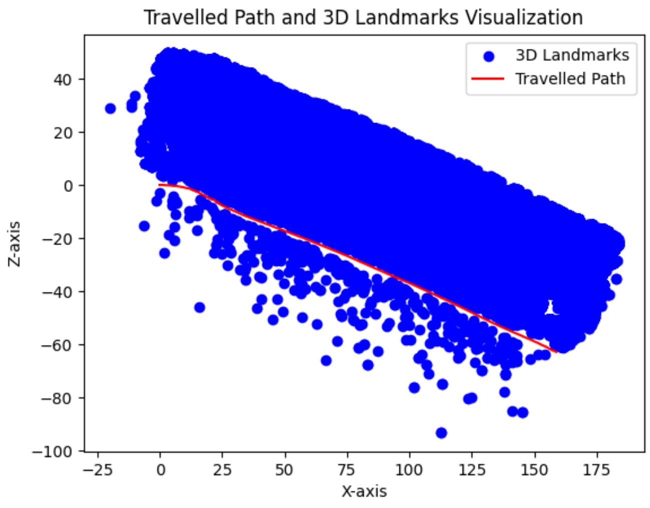
In this subsection, we visually present the results of the visual odometry system on the KITTI, MALAGA and parking datasets. Figure 4 shows the estimated trajectories for the full datasets. When comparing the estimated trajectories with the ground truth ones (see figure 5), the system demonstrates accurate and locally consistent pose estimation. A more detailed look is given in figure 6, where the local and global trajectory, together with the keypoints and 3D landmarks, is visualized for the first right turn in the KITTI dataset. Clear local consistency is shown. This image is part of a video showing the VO pipeline for the KITTI dataset. This video, as well as the videos for the MALAGA and parking dataset, can be found as attachments to this report.



(a) KITTI trajectory

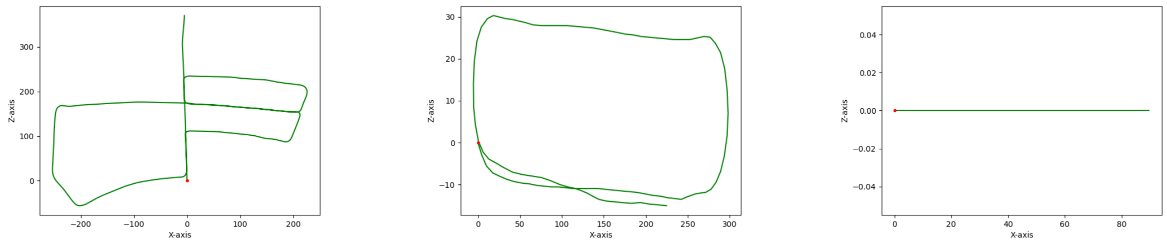


(b) MALAGA trajectory



(c) Parking trajectory

Figure 4: Estimated Trajectory and Triangulated 3D landmarks for the full KITTI, MALAGA and Parking datasets respectively.



(a) KITTI trajectory (b) MALAGA trajectory (c) Parking Trajectory

Figure 5: Ground Truth Trajectories for the KITTI, MALAGA and Parking datasets respectively.

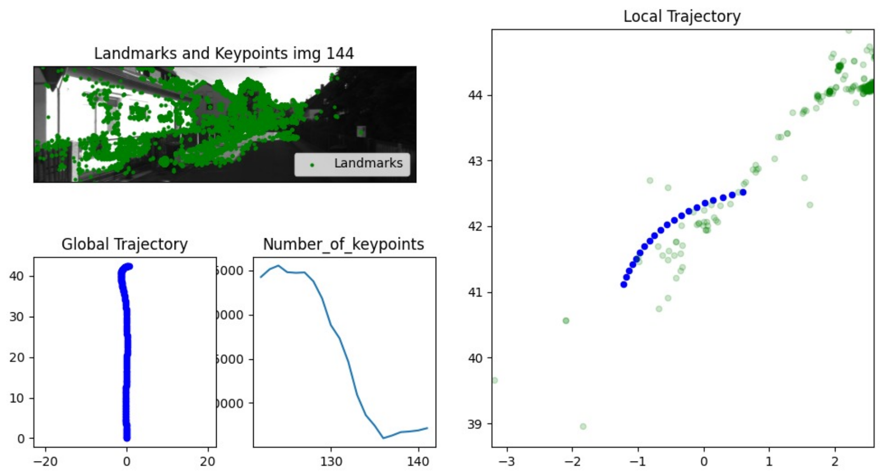


Figure 6: Screencast showcasing local and global trajectory, together with the keypoints and 3D landmarks, for the first right turn in the KITTI dataset. Clear local consistency is shown.

3.1 Discussion

A crucial system choice of the VO pipeline was how new candidate keypoints were added to the KLT tracker. After rigorous testing of different detection approaches (Shi-Tomasi or SIFT detector) and matching methods (SIFT descriptor brute force matching or the usage of a mask parameter), the best approach used the SIFT detector and SIFT descriptor brute force matching to add new candidate keypoints. Using SIFT in this step made the pipeline more robust compared to the other approaches.

After extensive parameter tuning, the following parameters were selected for the final system: 1. A RANSAC confidence score of 0.99999 for the pose estimation in the continuous operation (section 2.3) and 2. a threshold alpha (angle between the bearing vector of the first observation and the bearing vector of the current observation) of 1 degree to validate new keypoints and add them to the state (see section 2.4). During the development of the pipeline, we noticed that the initialization played an important role in the quality of the visual odometry. The quality of the initialization often played a more influential role than changing some of the parameters listed above. This was an interesting finding.

In the KITTI and MALAGA datasets, our VO pipeline often failed due to a drop in convertible candidate keypoints, leading to insufficient data (inliers) for the P3P algorithm in pose estimation. To address this, we implemented a keypoint threshold check. If the actual keypoints fell below 4, the system would bootstrap again, skipping 4 frames in the process. This approach ensured continuity and accuracy in pose estimation by maintaining a sufficient keypoint count.

3.2 Future Work

We identified 2 areas of future work: 1. making the pipeline faster (to be viable in real-time applications) and 2. making the pipeline globally consistent. The main bottleneck of the speed of the pipeline is the way how new candidate keypoints are added to the state right now.
In each frame, SIFT detection takes place to identify new keypoints. To make this process more efficient, we explored using the Shi-Tomasi detector, which gave less robust results. A direction of future work would be to add new keypoints only after e.g. 3 frames instead of every frame. This would speed up the system substantially.
To make the pipeline globally consistent, additional features such as bundle adjustment and loop closure should be added to the system.

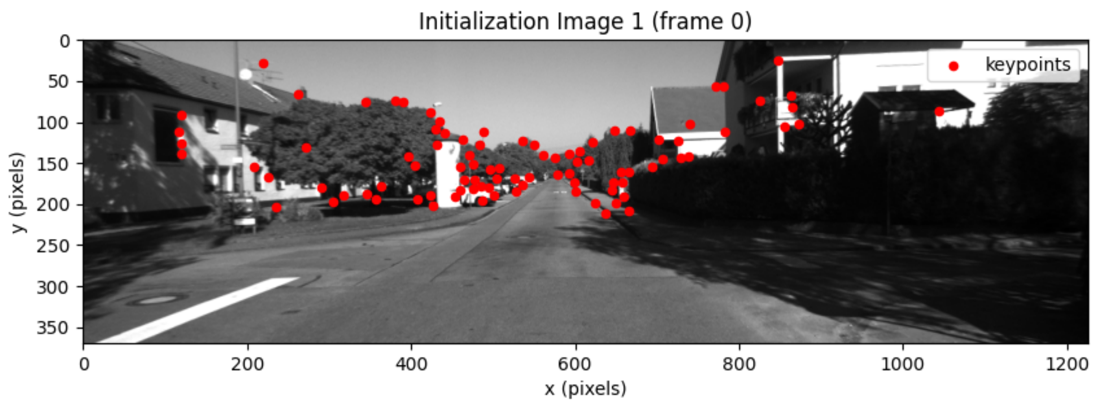
References

[1] José-Luis Blanco-Claraco, Francisco Ángel Moreno-Dueñas, and Javier González-Jiménez. The málaga urban dataset: High-rate stereo and lidar in a realistic urban scenario. *The International Journal of Robotics Research*, 33(2):207–214, 2014.

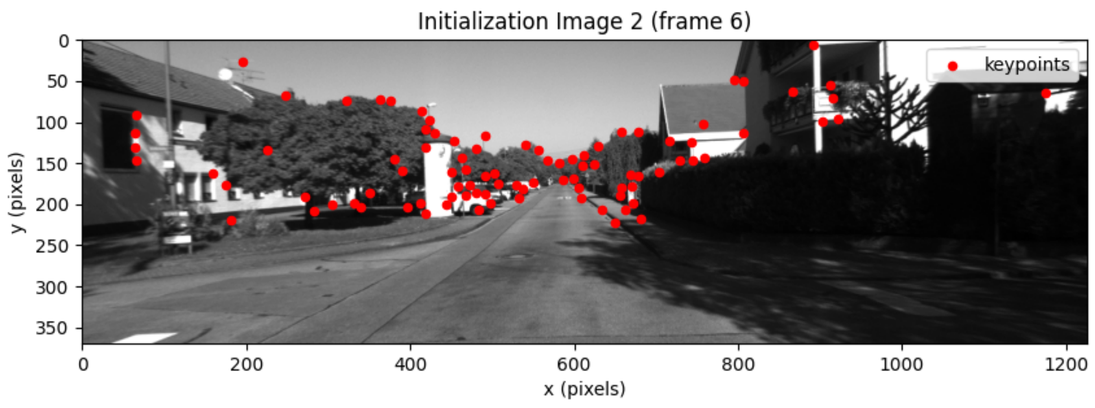
[2] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Appendix

A Initialization



(a) Shi-Tomasi Keypoint correspondences in the first initialization frame of the KITTI dataset.



(b) Shi-Tomasi Keypoint correspondences in the second initialization frame (06) of the KITTI dataset.

Figure 7: Shi-Tomasi Keypoint correspondences of the two initialization frames [0,6] of the KITTI dataset.