Politecnico di Torino

Master's Degree in Electronic Engineering

A.Y. 2024/2025

Efficient Deep Visual–Inertial Odometry for robot localization

Supervisors:
Prof. Marcello Chiaberge
Prof. Stefano Primatesta
Dott. Mauro Martini

Candidate:
Marco Pasculli

A nonno Peppe,
la mia guida

Table of Contents

List of Figures

Chapter 1

Introduction

Autonomous navigation is a fundamental capability for a wide range of robotic systems, from aerial drones and autonomous vehicles to mobile robots in general. A key component enabling such autonomy is the ability to accurately estimate the motion of the system over time, a task commonly referred to as *odometry*. Incorporating the estimated motion within a global frame or a map leads to *localization*. For this purpose, several techniques have been proposed over the years, each with complementary strengths and weaknesses, depending on the sensors used and the approach taken.

The standard solution for localization in a wide range of applications in outdoor robotics is Global Navigation Satellite Systems (GNSS), such as GPS or Galileo. These systems provide high-accuracy and reliable global positioning capabilities, especially in open-sky environments. However, GNSS-based localization suffers from an important limitation due to the fact that signals can become unreliable or entirely unavailable due to signal occlusion in environments such as indoor settings, urban canyons, dense forests or tunnels.

To overcome these challenges in GNSS-denied environments, alternative localization methods are required.
The most implemented type of odometry in these scenarios for wheeled robots is *wheel odometry*, which uses rotary encoders attached to the robot's wheels, to measure the rotation of each wheel. The encoder data combined with the robot's kinematic model, allows to estimate the robot's overall motion. Despite being the most common and simplest, it suffers from position drift due to wheel slippage, when the wheels lose traction and rotate without fully translating into robot movement, causing imprecision in the estimation of the translation.
In operations involving an environment that might cause this problem, other techniques are employed.

Visual Odometry (VO) estimates the pose of a robot using images acquired from single or multiple cameras attached to the robot and had become one of the most robust techniques for vehicle localization. It extracts motion-related information from the apparent displacement of visual features across frames. This can be achieved using geometric relationships between camera poses and 3D scene points. Despite their effectiveness, purely visual approaches have drawbacks such as motion blur, change in lighting or insufficient visual elements.
Another alternative is inertial navigation, that estimates a robot's position, velocity, and orientation by integrating data from accelerometers and gyroscopes. These sensors measure linear acceleration and angular velocity, respectively, allowing the system to track motion over time without relying on external signals. This is one of the main advantages of inertial navigation, making it suitable for GPS-denied or visually degraded environments. However, since because of the integration process even small sensor errors accumulate rapidly, inertial navigation systems (INS) suffer from drift over time.
Visual-Inertial Odometry (VIO) combines visual and inertial information to overcome these individual limitations, achieving a more robust and accurate motion estimation.
In recent years, *Deep Learning* has emerged as a powerful alternative to traditional visual odometry techniques. Data-driven models reduce the dependence on hand-engineered features and geometric models, learning robust representations directly from raw sensor data. This shift has led to the development of end-to-end and hybrid deep VO architectures, within a unified framework, that can perform feature extraction, data association, and pose estimation. Convolutional Neural Networks (CNNs) are widely used for learning visual features that are invariant to scale, rotation, and illumination changes. These features can be used to predict depth maps, estimate optical flow, or directly regress relative poses between frames.

1.1 Goal of the thesis

This thesis focuses on the development of a hybrid deep learning-based visual-inertial odometry system, aiming to leverage the representational power of convolutional neural networks and recurrent architectures to detect and match visual features, while fusing this information with inertial data by means of a direct Extended Kalman Filter (EKF). The objective is to design and implement a system capable of pose estimation, with potential applications in autonomous navigation for rover-like robots.

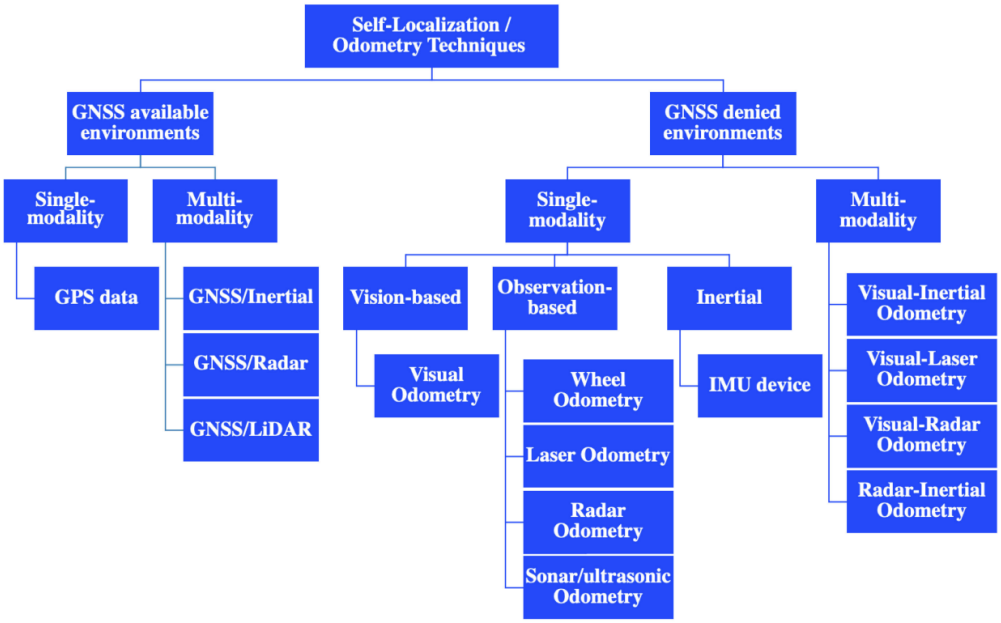


Figure 1.1: Self-localization odometry techniques [1]

1.2 Thesis structure

The thesis is organized as follows:

- **Chapter 2** introduces the existing methods and provides a literature review of the most influential research papers in the fields of traditional visual odometry, deep visual odometry and visual-inertial odometry.
- **Chapter 3** introduces the theoretical background of visual odometry and describes the methods and design choices adopted in the development of the algorithm proposed in the thesis.
- **Chapter 4** details the hardware setup used for data acquisition and system deployment.
- **Chapter 5** presents the tests setup with their relative experimental results and performance evaluations.
- **Chapter 6** concludes the thesis and discusses potential directions for future work.

Chapter 2

Literature Review

2.1 DeepVO: Towards end-to-end visual odometry with deep Recurrent Convolutional Neural Networks [2]

In a period when Deep Learning had not been much adopted for VO problems, since most CNNs were trained for recognition and classification, Wang et al. proposed a novel deep learning-based VO technique, whose architecture,that they refer to as Recurrent Convolutional Neural Networks(RCNNs), combines convolutional and recurrent layers.
The authors' motivation for developing a deep learning algorithm for VO stems from a critical observation: conventional VO pipelines, though effective in controlled conditions, require substantial manual fine-tuning and prior knowledge (e.g. camera height) to estimate absolute scale, and often fail to generalize across environments with different visual characteristics.
In contrast to classical methods, typically composed of discrete modules such as feature detection, matching, motion estimation and scale recovery, this work introduces a fully data-driven alternative that learns to estimate camera motion directly from sequences of raw RGB images.

2.1.1 Network architecture and workflow

At the heart of DeepVO there is a twofold architecture: a convolutional neural network (CNN) for feature extraction and a recurrent neural network (RNN), specifically a stack of Long Short-Term Memory (LSTM) units, for modeling sequential dependencies. At each time step, two consecutive RGB frames are preprocessed and then stacked together to form the input tensor. The CNN then

extracts the geometric features, which are subsequently passed to the LSTM to infer the 6-DoF pose of the camera.

Since, unlike networks designed for object classification, it focuses on learning geometric features instead of appearances or visual context, the CNN is tailored for motion understanding in unknown environments. Its structure, inspired by FlowNet, consists of nine convolutional layers, each followed by a rectifying linear unit (ReLU), with increasing channel depth and decreasing spatial resolution. This allows the network to abstract small and useful features.

The RNN part, composed of two Long Short-Term Memory (LSTM) layers in series, each with 1000 hidden units,thanks to its ability to model dependencies in a sequence, is responsible for keeping track of the temporal evolution of motion. However, RNN is not suitable to directly learn sequential representation from high-dimensional data, such as images, therefore, its input are the features previously exracted by the CNN. By maintaining internal memory of the previous states thanks to gating mechanisms and feedback loops, it is capable of finding relations among the CNN features, modeling the dynamics of the system.

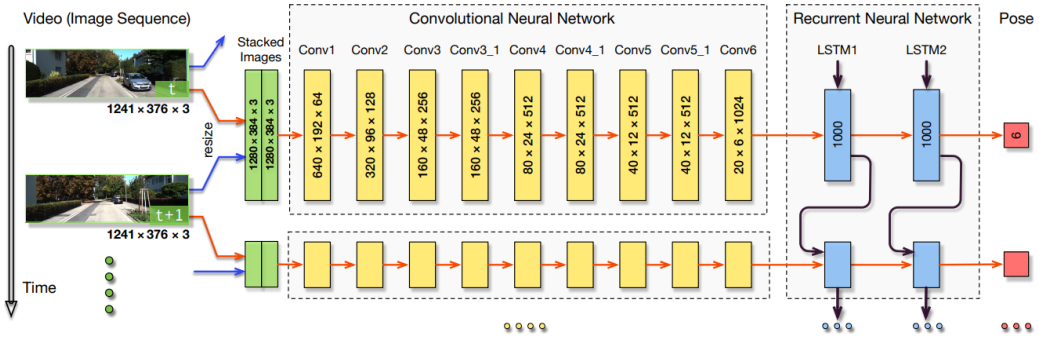


Figure 2.1: DeepVO network architecture

2.1.2 Loss function and training objective

The network is trained to minimize the euclidean distance between predicted poses and ground-truth poses. To this end, the loss function penalizes both translational and rotational errors using the Mean Square Error (MSE):

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{t} \|\hat{p}_k - p_k\|^2 + \kappa \|\hat{\varphi}_k - \varphi_k\|^2$$

where $\hat{p}_k$, $\hat{\varphi}_k$ denote the predicted position and orientation at time k, and p_k, φ_k are the ground truth values. The scalar κ is a weight that balances the two terms,

and N is the number of samples. In this computations the orientation is represented via Euler angles rather than quaternions to avoid optimization complications.

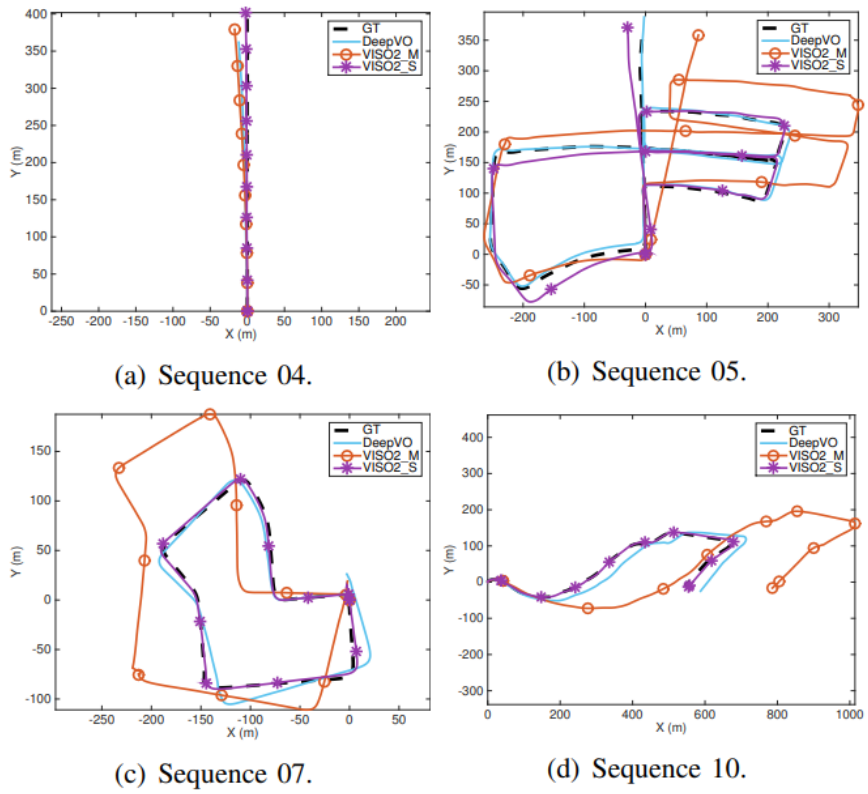
2.1.3 Performances and observations

Both training and testing of the DeepVO network have been performed on the well-known KITTI dataset. The results were compared with some established monocular and stereo systems such as VISO2. The results were promising: although stereo systems still have an edge in absolute accuracy, DeepVO significantly outperforms monocular ones. One particular result is that while during the tests in high speed scenarios the translational error increased, the orientation error decreased. This is probably due to the fact that the testing scenarios were mostly at low speed. Moreover, when a car goes at high speed, it tends to go straight and this explains the low orientation error.
Ultimately, the good results of the tests demonstrate the ability to maintain scale consistency without external cues, an impressive step in advance for monocular VO methods.

However, the paper also highlights challenges inherent to deep learning approaches. In particular, overfitting is identified as a major issue, especially for rotational components, which exhibit low variability and are therefore more easily memorized by the network. The experiments clearly show that well-regularized models generalize better to unseen sequences, a crucial property for real-world deployment.

2.1.4 Conclusions

DeepVO represents an important early step toward end-to-end learning in robot perception. While it does not aim to replace classical geometry-based VO outright, it offers a compelling alternative that simplifies the pipeline and removes the need for manual tuning. Its ability to learn both geometric representations and motion dynamics directly from data makes it particularly suitable for deployment in scenarios where environmental conditions vary or prior calibration is unavailable.



(a) Sequence 04. (b) Sequence 05.

(c) Sequence 07. (d) Sequence 10.

Figure 2.2: DeepVO testing results

2.2 ORB-SLAM: Feature-Based Monocular Visual Odometry and SLAM [3]

ORB-SLAM, proposed in 2015 by Mur-Artal et al., is one of the most influential and widely used monocular SLAM systems in the field of robotics and computer vision. Although its primary purpose is full SLAM (Simultaneous Localization and Mapping), the system can also operate in pure odometry mode, making it an excellent reference for evaluating monocular visual odometry methods. It demonstrates the capabilities and limitations of traditional feature-based pipelines: high accuracy when features are well distributed and trackable, but sensitivity to visual degradations or environments with few distinctive features.
One of the major advantages of ORB-SLAM is its robustness in long-term operation, thanks to techniques such as Bundle Adjustment (BA) and a so-called Essential Graph (EG), that is a real time loop closing based on the optimization of a pose graph. The algorithm works well also under challenging conditions such as dynamic objects, low texture, or lighting changes.

2.2.1 System overview

The architecture of ORB-SLAM is threefold: the threads that run in parallel are tracking, local mapping, and loop closing.
The tracking thread extracts ORB features from each frame and estimates the camera pose via feature matching and motion-only bundle adjustment (BA). If a sufficient number of features match an existing keyframe, the system attempts relocalization using a bag-of-words database.
The local mapping thread maintains a sparse 3D map using selected keyframes and performs local bundle adjustment to refine the structure and poses.
Finally, the loop closing thread detects previously visited places and performs pose-graph optimization followed by a full map optimization (essentially a form of global bundle adjustment), which helps reduce accumulated drift over time.
For the purpose of being a representative example of classical VO technique, only the tracking thread is described in detail in the next section.

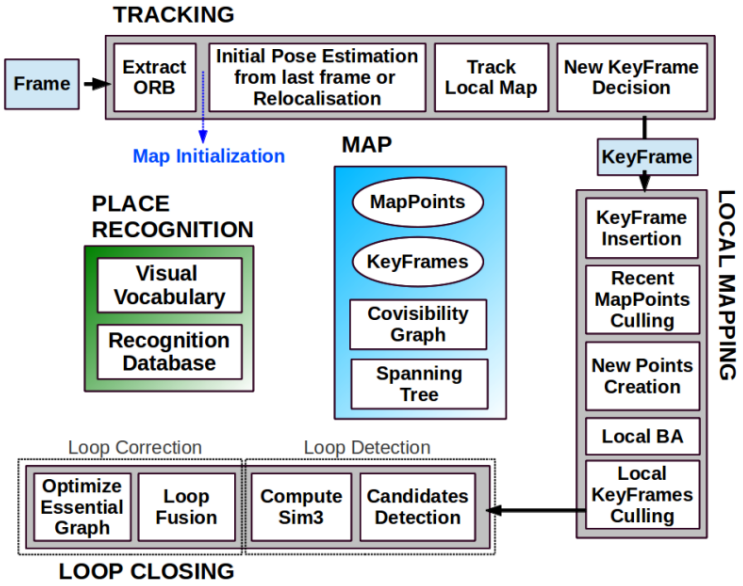


Figure 2.3: ORB Slam architecture

2.2.2 Tracking thread

The need for ORB-SLAM to run in real time rules out the well known SIFT and SURF from the choice of the feature extractors, since their computation time is approximately 300 ms. The authors opted to rely on ORB (Oriented FAST and Rotated BRIEF) features for all parts of the pipeline: tracking, mapping,

localization, and loop closure. This design choice enables the system to be efficient and consistent across different environments while also allowing it to operate in real time using only CPU resources.

In the tracking thread, FAST corners at 8 scale levels with a scale factor of 1.2 are extracted. For image resolutions below 752 $\times$ 480 pixels the number of corners to be extracted is set to 1000, while for higher resolutions, as the 1241 $\times$ 376 in the KITTI dataset, 2000 corners are extracted. In order to ensure an homogeneous distribution each image is divided in cells, trying to extract at least 5 corners per cell, adapting the detector threshold if not enough corners are found. The amount of corners retained per cell is also adapted if some cells contains no corners (textureless or low contrast). The ORB descriptor are then computed on the obtained FAST corners and used for feature matching.
If tracking for the previous frame was successful, a guided search of the map points seen in the previous frame is performed and the camera pose is computed using a constant velocity motion model. A broader search of the map points surrounding their position in the most recent frame is performed if there were not enough matches between the two consecutive frames.
In the event that tracking is lost, the frame is turned into a bag of words and keyframe candidates for global relocalization are looked for in the recognition database.Then correspondences with ORB associated to map points in each keyframe are computed and by means of the PnP algorithm a camera pose is found. If there were enough inliers, the pose is optimized and a guided search is conducted for additional matches with the candidate keyframe's map points.

2.2.3 Experimental results on localization performance

ORB-SLAM have been tested on several datasets, such as KITTI, TUM RGB-D, and EuRoC MAV. Among the results presented in the paper, the ones of most interest for this thesis are related to the localization tests with the KITTI dataset. The results are reported in the table below.

Table 2.1: Results of ORB-SLAM in the KITTI dataset.

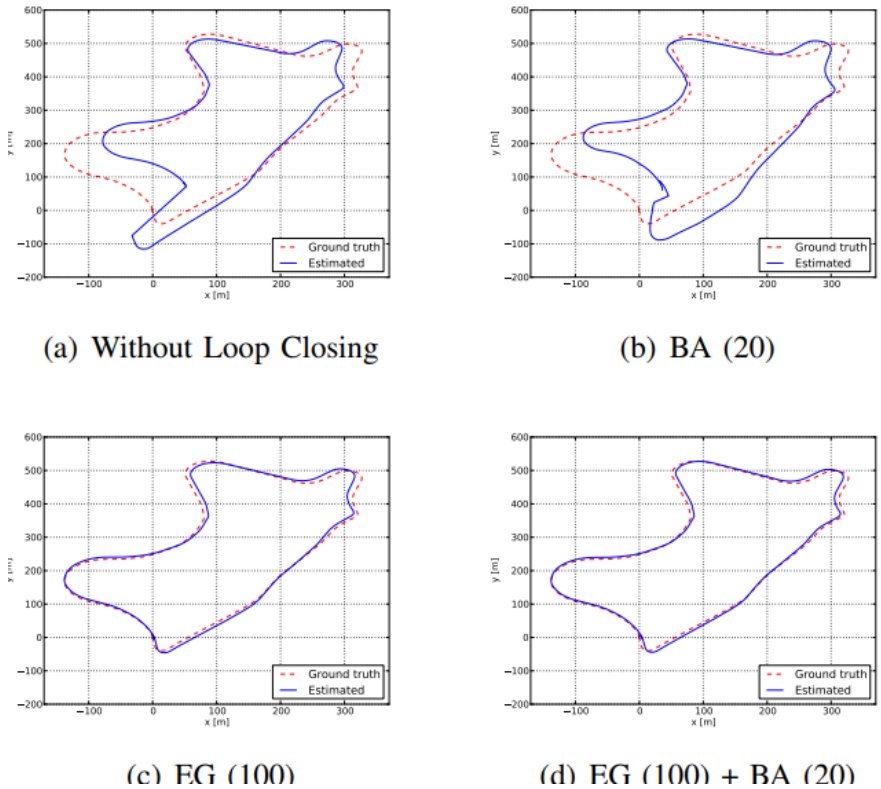
		ORB-SLAM		**Global BA (20 its.)**	
Sequence	Dimension (m×m)	KFs	RMSE (m)	RMSE (m)	Time BA (s)
KITTI 00	564 × 496	1391	6.68	5.33	24.83
KITTI 01	1157 × 1827	X	X	X	X
KITTI 02	599 × 946	1801	21.75	21.28	30.07
KITTI 03	471 × 199	250	1.59	1.51	4.88
KITTI 04	0.5 × 394	108	1.79	1.62	1.58
KITTI 05	479 × 426	820	8.23	4.85	15.20
KITTI 06	23 × 457	373	14.68	12.34	7.78
KITTI 07	191 × 209	351	3.36	2.26	6.28
KITTI 08	808 × 391	1473	46.58	46.68	25.60
KITTI 09	465 × 568	653	7.62	6.62	11.33
KITTI 10	671 × 177	411	8.68	8.80	7.64

ORB-SLAM achieved low drift rates for both translation and rotation, even though in long sequences without loop closure, the results are clearly worse. In particular, the system benefits from its keyframe-based design, which allows it to perform bundle adjustment at each frame, keeping the error even lower. The accuracy is particularly high in urban scenarios characterized by abundant static features, that facilitate the feature matching process.
For what regards the TUM RGB-D dataset, the authors report good accuracy even across sequences with challenging conditions. This is due to the quality of ORB features, which remain relatively invariant to illumination changes and moderate motion blur.
The EuRoC MAV dataset further demonstrates ORB-SLAM's capability in aerial robotics. Even without loop closure, the system is able to estimate a consistent trajectory in indoor spaces, where the field of view of the camera is often constrained. However, the authors note that in highly textureless environments, such as monocromatic corridors, the system's reliance on point features can lead to tracking loss.
Overall, the localization results confirm that ORB-SLAM delivers high robustness across a variety of scenarios. Loop closure has a crucial role in the accuracy of the results, as showed in the table and in the images below:

Table 2.2: Comparison of loop closing strategies in KITTI 09.

Method	Time (s)	Pose Graph Edges	RMSE (m)
-	-	-	48.77
BA (20)	14.64	-	49.90
BA (100)	72.16	-	18.82
EG (200)	0.38	890	8.84
EG (100)	0.48	1979	8.36
EG (50)	0.59	3583	8.95
EG (15)	0.94	6663	8.88
EG (100) + BA (20)	13.40	1979	7.22

Essential Graph has the most influence on the accuracy of the results, with its combination with bundle adjustment results in the lowest RMSE value.



(a) Without Loop Closing (b) BA (20)

(c) EG (100) (d) EG (100) + BA (20)

Figure 2.4: Comparison of different loop closing strategies in KITTI 09

2.3 VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State [4]

Proposed by Qin, Li, and Shen in 2018, VINS-Mono is a tightly-coupled, optimization-based visual-inertial odometry system. It consists of a monocular camera and an inertial measurement unit (IMU), which is the minimum sensor suite in size, weight, and power.
Unlike filter-based methods such as the Multi-State Constraint Kalman Filter (MSCKF), VINS-Mono uses non linear optimization to fuse visual and inertial data with a sliding window approach, allowing for accurate state estimation by jointly optimizing feature positions and IMU measurements.
An important aspect of the system is that it can estimate the absolute scale of motion directly, thanks to the integration of inertial data. This overcomes one of the main limitations of monocular VO systems, which typically suffer from scale ambiguity unless external cues are provided.

2.3.1 System architecture

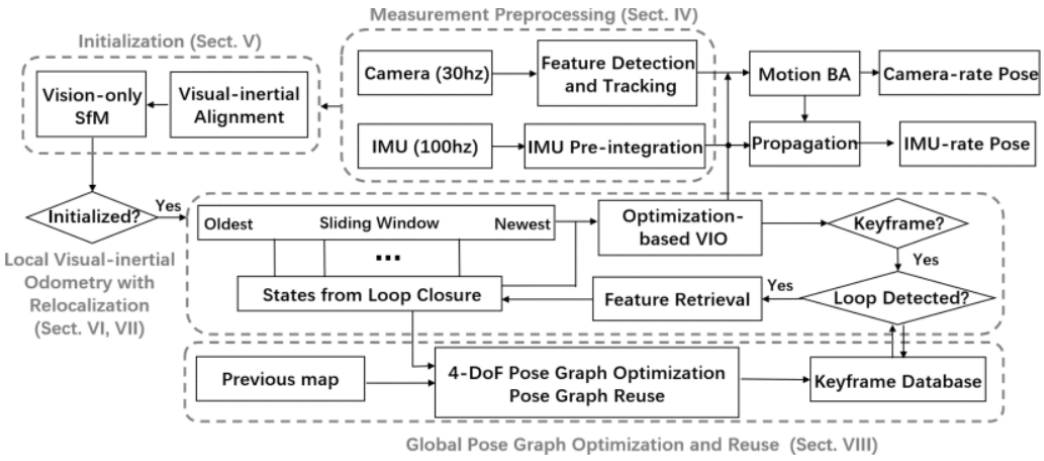


Figure 2.5: VINS-Mono architecture

The VINS-Mono pipeline consists of several interconnected modules:

- **Feature tracking:** The KLT sparse optical flow technique tracks existing features for every new image [5]. In the meantime, to keep each image's feature count at a minimum of 100–300, new corner features are identified. By establishing a minimum pixel spacing between two adjacent features, the detector ensures a consistent feature distribution. After passing outlier

rejection, 2D features are first undistorted and then projected onto a unit sphere. Using a basic matrix model and RANSAC, outlier rejection is carried out.
Keyframes are also selected in this step, by means of two cryteria. The first is to compute the average parallax apart from the previous keyframe, checking wether it sets above a certain threshold. The second is to count the number of features that have been tracked from the previous frame, checking that this number is below a certain value.

- **IMU pre-integration:** The authors use the continuous-time quaternion-based derivation of IMU preintegration. Acceleration bias (ba), gyroscope bias (bw), and additive noise all have an impact on IMU measurements, which are taken in the body frame and integrate the force for fighting gravity with the platform dynamics. ŵ and â, the raw accelerometer and gyroscope readings, are provided by:

$$\begin{aligned} \hat{\mathbf{a}}_t &= \mathbf{a}_t + \mathbf{b}_{a_t} + \mathbf{R}_w^t \mathbf{g}^w + \mathbf{n}_a \\ \hat{\boldsymbol{\omega}}_t &= \boldsymbol{\omega}_t + \mathbf{b}_{w_t} + \mathbf{n}_w. \end{aligned} \tag{1}$$

The biases are modeled as random walks and the additive noises in the measuraments are considered gaussian withe noises.
All the IMU measuraments between two frames are preintegrated following:

$$\begin{aligned} \boldsymbol{\alpha}_{b_{k+1}}^{b_k} &= \iint_{t\in[t_k,t_{k+1}]} \mathbf{R}_t^{b_k}(\hat{\mathbf{a}}_t - \mathbf{b}_{a_t})dt^2 \\ \boldsymbol{\beta}_{b_{k+1}}^{b_k} &= \int_{t\in[t_k,t_{k+1}]} \mathbf{R}_t^{b_k}(\hat{\mathbf{a}}_t - \mathbf{b}_{a_t})dt \\ \boldsymbol{\gamma}_{b_{k+1}}^{b_k} &= \int_{t\in[t_k,t_{k+1}]} \frac{1}{2}\boldsymbol{\Omega}(\hat{\boldsymbol{\omega}}_t - \mathbf{b}_{w_t})\boldsymbol{\gamma}_t^{b_k} dt \end{aligned} \tag{3}$$

- **Sliding window optimization:** At the core of the system lies a nonlinear optimization running over a sliding window of keyframes. The state vector includes the poses of the camera/IMU, velocity, biases, and selected landmark positions. The cost function integrates both visual reprojection errors and IMU pre-integration residuals. By solving this optimization problem, the system achieves tightly-coupled sensor fusion: the vision constraints help to bound the inertial drift, while the inertial constraints provide metric scale and aid visual tracking during periods of poor feature visibility.
To maintain computational tractability, the sliding window is kept fixed in size, typically containing around ten keyframes. When new frames arrive, older states are marginalized out using the Schur complement, while their information is preserved in the form of prior constraints. This strategy ensures

that the system can run in real time while retaining a sufficient temporal horizon to achieve accurate estimation.

- **Loop closure and relocalization (optional):** Beyond local odometry, VINS-Mono integrates global optimization capabilities. A loop closure module based on a bag-of-words approach detects revisited locations. When a loop closure is identified, the system introduces additional constraints in a pose graph, which is then optimized to reduce drift accumulated over long trajectories. The relocalization mechanism also allows the system to recover from tracking failures, making it more robust to temporary occlusions. In this way, VINS-Mono can function both as a visual-inertial odometry system with bounded drift and as a full SLAM system with global consistency. .

2.3.2 Experimental validation

The authors validated VINS-Mono extensively across a wide range of datasets and platforms, performing numerical analysis to define the accuracy of the system.
On the EuRoC MAV dataset, VINS with and without loop closure activated have been compared to another state of the art algorithm for VIO, OKVIS []. The tests show that the best results are achieved by the VINS with loop closure algorithm.

Table 2.3: RMSE in EuRoC datasets in meters.

Sequence	OKVIS (mono)	VINS-Mono	VINS-Mono + Loop
MH_01_easy	0.232	0.212	0.212
MH_02_easy	0.175	0.169	0.163
MH_03_medium	0.307	0.227	0.120
MH_04_difficult	0.457	0.436	0.190
MH_05_difficult	0.697	0.558	0.198
V1_01_easy	0.083	0.078	0.078
V1_02_medium	0.074	0.072	0.071
V1_03_difficult	0.079	0.072	0.072
V2_01_easy	0.084	0.077	0.077
V2_02_medium	0.076	0.073	0.073
V2_03_difficult	0.074	0.072	0.072

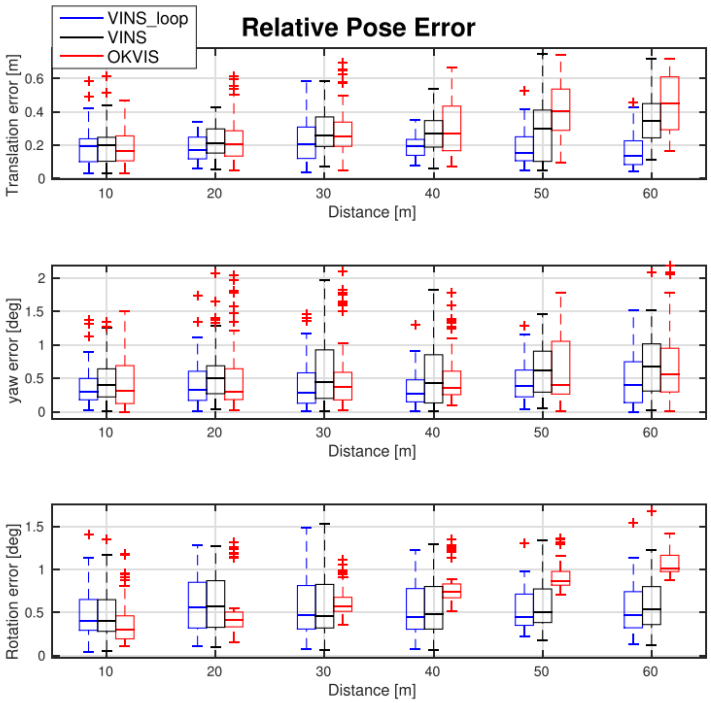


Figure 2.6: Relative pose error. Three plots are relative errors in translation, yaw, and rotation, respectively

In addition to benchmark datasets, the system was tested in real-world applications, including deployment on drones. These experiments demonstrated that VINS-Mono is not only accurate but also computationally efficient, capable of running in real time.

Chapter 3

Algorithm implementation

As described in the Introduction section, there exist many ways to implement a system capable of keeping track of a robot's position, given a certain sensor suite. The work of this thesis focuses on the development of a hybrid deep learning-based stereo visual-inertial odometry system that leverages a direct Extended Kalman Filter (EKF) to fuse the data from the visual and inertial sensors.
In this chapter, the implementation of the system under test is described in detail, focusing on the methodologies applied in the two main parts of the algorithm: the VO and the EKF.

3.1 Visual Odometry: methodology

The position of a mobile robot with vision-based odometry can generally be estimated in three different ways: through a feature-based approach, an appearance-based approach or a hybrid-based approaches.
The appearance-based approach estimates the camera pose by analyzing the intensity of the captured image pixels based on minimizing the photometric error, that is the difference between the intensity values of image pixels in corresponding areas of different images. The fundamental assumption in appearance-based methods is that the scene's appearance remains constant over the short time interval between two frames. This indicates that camera movement, not modifications to the scene, is the cause of any change in the perceived intensity of a pixel. The most widely used appearance-based technique is called Optical Flow (OF), which estimates mobility by processing raw pixel data from successive frames using an OF algorithm. This program examines variations in pixel intensity between two consecutive camera-captured frames. Calculating the 2D displacement vectors of points between frames—which represent the motion of objects and the camera—is the main principle. As the illumination of pixels changes, the OF algorithm tracks

these changes to determine how each pixel's position shifts from one frame to the next, thereby estimating the overall camera motion.
Since the appearance-based methods work on the assumption that the projection of a point in both frames has the same intensity. This assumption often fails due to lighting changes, sensor noise, pose errors and dynamic objects. Another issue is high computation due to the use of all pixels over all frames.
A feature-based technique was chosen as the VO algorithm for this thesis study because of those issues.
Overall, the basic algorithm for feature-based VO involves detecting and tracking features in consecutive camera frames, estimating the camera motion using the correspondences between the features, estimating the 3D positions of the features using triangulation, and estimating the camera trajectory using the estimated motion and 3D positions [6].

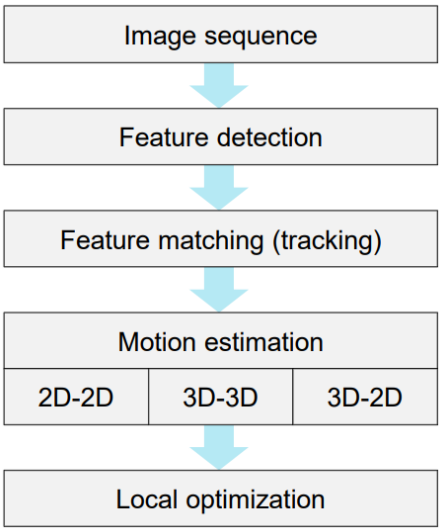


Figure 3.1: Traditional VO pipeline [7]

In this thesis work, a different approach is adopted: the tasks of feature detection and feature matching are assigned to two different Convolutional Neural Networks, namely SuperPoint [8] and SuperGlue [9], whose architecture is described in the following sections. Then the motion estimation is performed leveraging the information about the features matched in every consecutive frame couples, by means of a least squares optimization.

3.1.1 Mathematical formulation of the problem

In case of a stereo camera, the left and right frames are both captured at every discrete time instant, creating two sets of images: $I_{l,0:n} = \{I_{l,0}, ..., I_{l,n}\}$ and $I_{r,0:n} = \{I_{r,0}, ..., I_{r,n}\}$.
For simplicity, the left camera frame, the IMU frame and the robot frame are considered to be coincident. The trasformation between left and right camera $\mathbf{T}_r^l$ is instead known, and it's an important parameter for the motion estimation phase. Two camera positions at consecutive time instants $k-1$ and k are related by the rigid transformation $T_{k,k-1} \in \mathbb{R}^{4\times4}$: $T_{k,k-1} = \begin{bmatrix} \mathbf{R}_{k-1}^k & \mathbf{t}_{k-1}^k \\ \mathbf{0}^\top & 1 \end{bmatrix}$ where $\mathbf{R}_{k-1}^k \in SO(3)$ is the rotation matrix and $\mathbf{t}_{k-1}^k \in \mathbb{R}^{3\times1}$ is the translation vector. Known the initial position of the robot to be tracked, from the concatenation of all the consecutive transformations T_k, it's possible to obtain the consecutive camera poses $C_{0:n} = \{C_0, ..., C_n\}$, that is the trajectory of the robot. This means that the path is recovered incrementally, pose after pose, leading to possible drift after a long series of images: the uncertainty of the camera pose at time k is the combination of the uncertainty of the camera pose at time $k-1$ and the uncertainty of the transformation $T_{k,k-1}$.

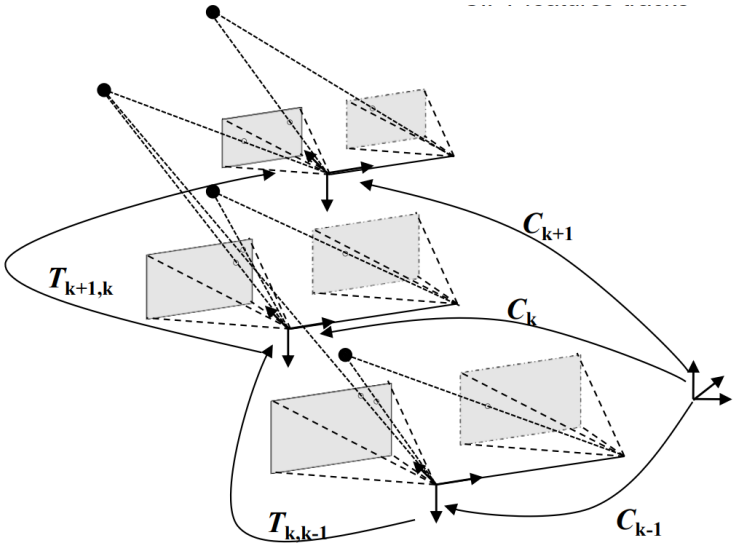


Figure 3.2: Matehmeatical Formulation [7]

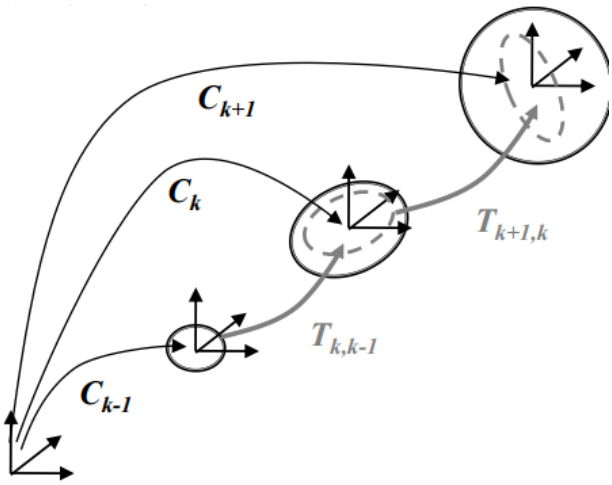


Figure 3.3: Uncertainty Propagation [7]

3.1.2 Feature detection

The first stage of the developed VO algorithm is *feature detection* and it is carried out by a neural network called SuperPoint.
"SuperPoint has a fully-convolutional neural network architecture, which operates on a full-sized image and produces interest point detections accompanied by fixed length descriptors in a single forward pass" [8]. The data flow is the following: first the input image dimensionality ($H \times W$) is processed and reduced by a factor 1/8 by the model using a single, shared encoder. The encoder applies a sequence of convolutional layers with non-linear activations and three max-pooling operations. This results in a feature tensor $B \in \mathbb{R}^{H/8 \times W/8 \times F}$. Each element of this tensor corresponds to a non-overlapping 8×8 patch of the original image, often referred to as a *cell*. This compact yet expressive representation is then shared by two decoder heads, that are in series to the encoder. Those decoders have two roles: one for feature detection and the other for feature description, both learning task-specific weights. The interest point decoder estimates the likelihood of a feature to be present in each cell. Specifically, it produces an output tensor of size $H_c \times W_c \times 65$, where the 65 channels correspond to the 64 pixel locations inside an 8×8 cell, plus one additional "dustbin" channel representing the absence of a keypoint.A channel-wise softmax is then applied, followed by a reshaping step that maps the tensor back to the original resolution $H \times W$, thus yielding a dense probability heatmap of interest points. This design avoids expensive upsampling operations and keeps the computation lightweight.
The second head computes descriptors associated with the detected points, generating a semi-dense grid of descriptors $D \in \mathbb{R}^{H_c \times W_c \times 256}$. Each descriptor corresponds

to a cell in the reduced resolution feature map. To obtain descriptors at the full image resolution, bicubic interpolation is applied, followed by ℓ_2 normalization to ensure unit-length descriptors:

$$\|\mathbf{d}_i\|_2 = 1 \quad \forall i.$$

This results in a dense map of descriptors $\in \mathbb{R}^{H \times W \times 256}$. Now, the descriptors corresponding to the detected keypoint locations can be put in output, waiting to be matched.
In contrast to traditional systems, which compute descriptors after detecting interest points and are unable to share computation and representation between the two tasks, the majority of the network's parameters are shared between the two tasks.

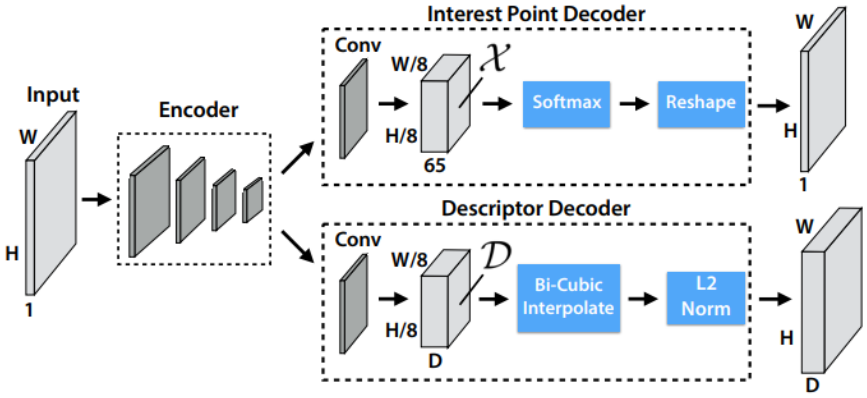


Figure 3.4: Superpoint Architecture [8]

The training of SuperPoint has been performed using pairs of synthetically warped images which have both pseudo-ground truth interest point locations and the ground truth correspondence from a randomly generated homography which relates the two images.
The final training objective combines two terms:

- **Detector loss** L_p, a cross-entropy loss over the probability map of interest points.
- **Descriptor loss** L_d, a hinge-based loss applied to pairs of descriptors. Positive pairs are enforced to have high similarity, while negative pairs are pushed apart, using margins m_p and m_n.

The global loss is:

$$L = L_p(X, Y) + L_p(X', Y') + \lambda L_d(D, D', S),$$

where S encodes ground-truth correspondences induced by a synthetic homography, and λ is a balancing factor.

3.1.3 Feature matching

Once the keypoints are detected and the descriptors are created, they have to be matched. In this work, the task of *feature matching* has been assigned to a neural network called SuperGlue.
SuperGlue has a three-stage pipeline: descriptors are first enriched with positional information thanks to a keypoint encoder, then refined through alternating self- and cross-attention layers, and finally matched via a differentiable optimal transport layer, supervised by a correspondence-aware cross-entropy loss.

The task is formulated as matching two sets of keypoints extracted from a pair of images. For each image, the input is a set of keypoints, each represented by its pixel location $\mathbf{x} \in \mathbb{R}^2$, a descriptor vector $\mathbf{d} \in \mathbb{R}^D$, and a detection score $s \in [0,1]$. The objective is to predict a partial assignment between these two sets while also accounting for unmatched keypoints due to occlusion or lack of overlap. Formally, the output is a binary matching matrix $\mathbf{M} \in \{0,1\}^{(N+1)\times(M+1)}$, where N and M are the number of keypoints in the two images and the additional row and column correspond to a dustbin of unmatched cases. The one-to-one constraint ensures that each keypoint is either matched to exactly one counterpart or left unmatched.

Descriptors by themselves are often insufficient for robust correspondence, since they lack positional awareness and information about detector confidence. SuperGlue addresses this with a *keypoint encoder*. Each descriptor $\mathbf{d}_i$ is projected into a latent space and augmented with a positional embedding computed from the normalized coordinates $\mathbf{x}_i$ and detection score s_i. Concretely:

$$\mathbf{f}_i = \mathbf{W}\mathbf{d}_i + \phi_{\text{pos}}([\mathbf{x}_i, s_i]),$$

where ϕ_{pos} is a multi-layer perceptron (MLP) and $\mathbf{W}$ is a linear projection. This step ensures that features carry both appearance and spatial context before entering the main graph neural network.

The central component of the architecture is an attentional graph neural network (GNN) that iteratively refines descriptors through message passing. Each layer consists of two operations:

- **Self-Attention.** Within each image, descriptors attend to one another, allowing features to capture intra-image relationships such as repetitive patterns. For a set of features $\mathbf{F} \in \mathbb{R}^{N\times D}$, the attention mechanism computes:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}} + \mathbf{B}\right)\mathbf{V},$$

where $\mathbf{Q} = \mathbf{F}\mathbf{W}_Q$, $\mathbf{K} = \mathbf{F}\mathbf{W}_K$, $\mathbf{V} = \mathbf{F}\mathbf{W}_V$ are query, key, and value projections, and $\mathbf{B}$ is a learnable bias function of the relative keypoint coordinates. This bias makes attention geometry-aware, steering it toward spatially meaningful relationships.

- **Cross-Attention.** To exchange information across the two images, each keypoint in one set attends to descriptors in the other set, conditioned on spatial offsets between coordinates. This enables the network to hypothesize potential correspondences and refine them iteratively. The formulation mirrors self-attention, but with $\mathbf{Q}$ from one image and $\mathbf{K}, \mathbf{V}$ from the other.

These two operations alternate in the network, allowing features to consolidate local structure before receiving cross-view evidence, and vice versa. At the output of this part of the network, the descriptors are no longer purely local but are enriched with both intra- and inter-image context.

Once descriptors are updated, SuperGlue computes a similarity matrix S_{ij}. To allow for unmatched points, the matrix is extended with an extra row and column, the dustbin. SuperGlue then solves for a soft assignment matrix $\mathbf{P}$ using the Sinkhorn algorithm, which iteratively normalizes rows and columns to produce a doubly-stochastic matrix. This guarantees approximate one-to-one assignments, with additional probability mass assigned to the dustbin for outliers. At inference time, hard correspondences are obtained by taking the mutual argmax in $\mathbf{P}$, discarding those assigned to the dustbin.

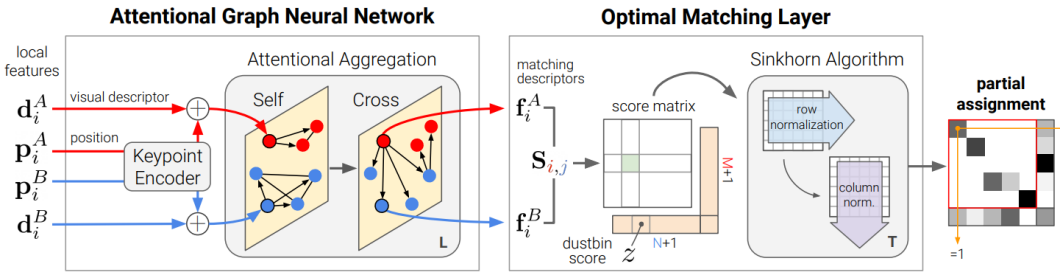


Figure 3.5: SuperGlue Architecture [9]

SuperGlue is trained with supervision on ground-truth correspondences. The loss is a cross-entropy over the soft assignment probabilities:

$$\mathcal{L} = -\sum_{(i,j)\in\mathcal{G}} \log \mathbf{P}_{ij} - \sum_{i\notin\pi_A(\mathcal{G})} \log \mathbf{P}_{i,\emptyset} - \sum_{j\notin\pi_B(\mathcal{G})} \log \mathbf{P}_{\emptyset,j},$$

where $\mathcal{G}$ is the set of ground-truth matches, and $\pi_A(\mathcal{G}), \pi_B(\mathcal{G})$ are the sets of indices that participate in them. The additional terms for unmatched keypoints

enforce that points without correspondences are correctly assigned to the dustbin. This loss makes the model robust to occlusions, low overlap, and repeated structures.

In figure 3.6, a demonstrative example of feature detection and matching by SuperPoint and SuperGlue from one of the tests performed is shown:

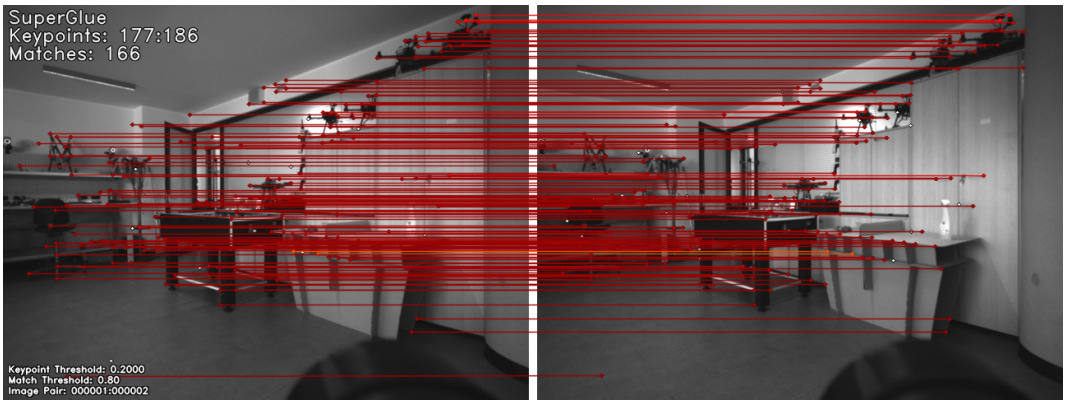


Figure 3.6: Example of feature detection and matching

3.1.4 Motion estimation

The last phase of the Visual Odometry algorithm is *motion estimation*. Motion estimation allows for the transformation matrix $T_{k,k-1}$ to be obtained, starting from two sets of features matched by SuperGlue between two consecutive left images. In general, motion estimation can be obtained in different ways from geometric constraints, depending on how the features that have been extracted and matched in the two previous phases are specified:

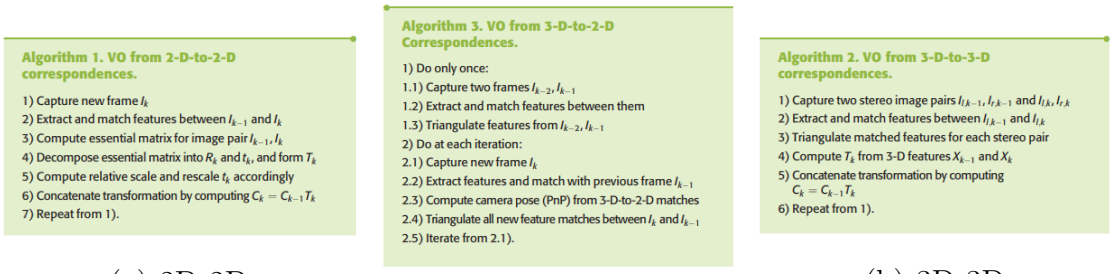
- 2D-2D: this case is typical of monocular systems. The transformation matrix $T_{k-1,k}$ can be estimated starting from the essential matrix $E_k = const \times t_k R_k$, up to an unknown scale factor. The main property of 2D-2D based motion estimation is the epipolar constraint. It determines the line on which, the feature corresponding to one in the first image, lies in the second image. This constraint can be formulated by $\tilde{p}'^{\top} E \tilde{p} = 0$, where $\tilde{p}'$ is a feature location in one image and $\tilde{p}$ is the location of its corresponding feature in another image. The minimal case solution for finding the essential matrix involves 5 couples of matched features.
  Starting from the essential matrix, 4 couples of R_k and t_k can be obtained: the correct one has all the features in front of both camera views. To find it, it's necessary to triangulate one of the features, checking its depth dimension.

- 3D-3D: this case is typical of stereo systems. The transformation matrix $T_{k-1,k}$ can be found by determining the transformation that aligns the two 3D feature sets. The general solution consists of finding the $T_{k-1,k}$ that minimizes the L_2 distance between the two 3-D feature sets. Of course, this method needs for the features to be triangulated from the two cameras perspectives.

- 3D-2D: the general formulation in this case is to find Tk that minimizes the image reprojection error:

$$\arg\min_{T_k} \sum_{i=1}^{N} \left\| \mathbf{p}_k^i - \hat{\mathbf{p}}_{k-1}^i \right\|^2$$

  where $\hat{\mathbf{p}}_{k-1}^i$ is the reprojection of the 3D point X_{k-1}^i into image I_k according to the transformation $T_{k-1,k}$ under test. The name of the problem is "Perspective from n Points (PnP)". In the algorithm proposed in this thesis work, the problem is solved by means of least square optimization starting from $n > 6$ couples of features.



(a) 2D-2D (c) 3D-2D (b) 3D-3D

Figure 3.7: Summary of the three algorithms for motion estimation [7]

3.2 EKF implementation

3.2.1 Theory on KF and EKF

In VIO systems, the fusion of data from both cameras and IMU sensors allows to achieve accurate and robust motion estimation. One of the most established methods for fusing sensor data is through the Kalman Filter (KF) and its extended version.
"In statistics and control theory, Kalman filtering (also known as linear quadratic estimation) is an algorithm that uses a series of measurements observed over time, including statistical noise and other inaccuracies, to produce estimates of unknown variables that tend to be more accurate than those based on a single measurement, by estimating a joint probability distribution over the variables for each time-step" [10].
The algorithm operates in two stages recursively: prediction and update. During the prediction phase the KF generates estimates of the current state variables, along with their uncertainty. These estimates are updated using a weighted average once the results of the subsequent measurement are known. Estimates with lower uncertainty are given more weight. With just the current input measurements, the previously computed state and its uncertainty matrix, the algorithm can function without the need for any other historical data.

The important variables and functions in this algorithm are:

- State: x_t
- Measurement: z_t
- Command: u_t
- State estimation function: $g(\cdot)$
- Measurement estimation function: $h(\cdot)$
- Mean of the gaussian estimating the state: μ_t
- Covariance of the gaussian estimating the state: Σ_t
- Motion model uncertainty: R_t
- Measurement model uncertainty: Q_t
- Jacobian of motion model with respect to the command: V_t
- Jacobian of motion model with respect to the state: G_t

- Jacobian of sensor model with respect to the state: H_t
- Kalman Gain: K_t

The motion model and sensor model uncertainties are a representation of the noise that is added to the estimate after each step. Those matrices can be difficult to tune and may vary for each application. The values in the motion model uncertainty matrix should be smaller when the model matches the system well, and this would be ideal. If however, the predictions from the motion model are far from the measurements, those values must be increased. This will cause the filter's predicted error to be larger, which in turn will cause the filter to trust the incoming measurement more during the correction step. The initial estimate covariance matrix, on the other hand, represents the initial value for the state estimate error. Setting the values in this matrix small, will result in rapid convergence for the initial measurement. However, care should be taken not to use large values for variables that will not be measured directly.
In the context of this thesis, these matrices were carefully tuned to ensure accurate state estimation. Given the accuracy of the VO algorithm and the noise concerning the IMU measurements, the variances of the motion model have been set to pretty high values, which are $3m/s^2$ (linear acceleration) and 2^o (angular velocity) while for the sensor model, the variance has been set to $0.001m$, a low value. By carefully tuning these parameters, it was possible to achieve a balance between rapid convergence of the state estimates and the accuracy of these estimates.

The Extended Kalman filter is the nonlinear version of the Kalman filter that linearizes the system about an estimate of the current mean and covariance.
The two fundamental equations of next state and the measurement estimations, are represented by nonlinear functions, respectively:

$$x_t = g(x_{t-1}, u_t)$$
$$z_t = h(x_t)$$

Those non-linear functions distort Gaussian distributions, resulting in totally different forms (figure 3.8).

In this scenario, the standard KF can be applied if the system is linearized at each time step. The EKF achieves this, by linearizing the system applying a first-order Taylor expansion to the functions $g(\cdot)$ and $h(\cdot)$ around the mean. The linearization approximates g and h by a linear function that is tangent to them at the mean of the Gaussian. Projecting the Gaussian through this linear approximation, keeps the posterior a Gaussians. In fact, once g and h are linearized, the mechanics of belief propagation are equivalent to those of the KF (Figure 3.9).

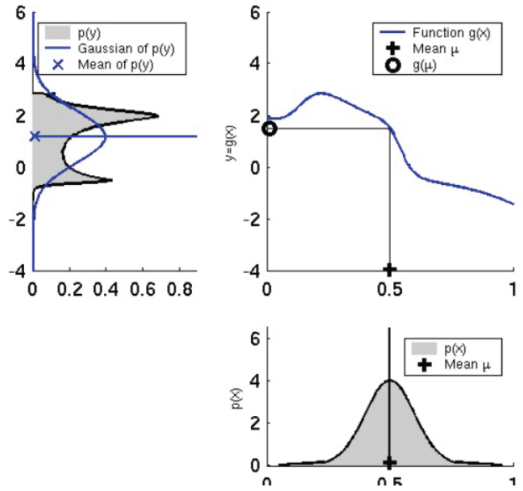


Figure 3.8: Non linear transformation of a gaussian

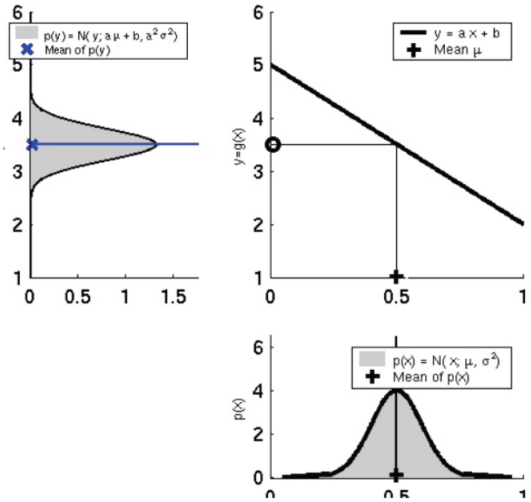


Figure 3.9: Linear transformation of a gaussian

The linearization of the state transition function g(.) is the following:

$$\begin{aligned} g(x_t, u_t) &\approx g(\mu_{t-1}, u_{t-1}) + \left.\frac{\partial g}{\partial x}\right|_{x_{t-1}=\mu_{t-1},\, u_{t-1}} (x_{t-1} - \mu_{t-1}) \\ &= g(\mu_{t-1}, u_{t-1}) + G_{t-1}\,(x_{t-1} - \mu_{t-1}) \end{aligned}$$

where the Jacobian G_t has size $n \times n$, with n denoting the state dimension. The value of the Jacobian depends on u_t and μ_{t-1}, hence it differs for different points in time.

The same linearization is implemented for the measurement function h. Here, the Taylor expansion is developed around $\bar{\mu}_t$, the state considered most probable when h is linearized.

$$h(x_t) \approx h(\bar{\mu}_t) + \left.\frac{\partial h}{\partial x}\right|_{x_t=\bar{\mu}_t} (x_t - \bar{\mu}_t) = h(\bar{\mu}_t) + H_t(x_t - \bar{\mu}_t)$$

Concerning the KF nad EKF algorithms, the main differences are indeed the state estimation function and measurement estimation function, which are linear in the KF and non-linear in EKF.
The two algorithms are reported in parallel in the following table:

Extended Kalman filter	**Kalman filter**
Prediction	
$\bar{\mu}_t = g(\mu_{t-1}, u_t)$	$\mu_t = A_t\mu_{t-1} + B_t u_t$
$\bar{\Sigma}_t = G_t\Sigma_{t-1}G_t^\top + R_t$	$\Sigma_t = A_t\Sigma_{t-1}A_t^\top + R_t$
Correction	
$K_t = \bar{\Sigma}_t H_t^\top (H_t\bar{\Sigma}_t H_t^\top + Q_t)^{-1}$	$K_t = \Sigma_t C_t^\top (C_t\Sigma_t C_t^\top + Q_t)^{-1}$
$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$	$\mu_t = \bar{\mu}_t + K_t(z_t - C_t\bar{\mu}_t)$
$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$	$\Sigma_t = (I - K_t C_t)\Sigma_t$
Return μ_t, Σ_t	

Regarding the computational efficiency of the EKF, each update has a complexity of $O(k^{2.8} + n^2)$, where k represents the size of the measurement vector z_t and n the size of the state vector x_t. This makes it very efficient, especially because it approximates the probability distribution with a multivariate Gaussian. One crucial limitation of the EKF is due to its approximation of state transitions and measurements model by linear Taylor expansions. The accuracy of this approximation depends mainly on two factors: the degree of nonlinearity of the functions and the level of uncertainty, that means the width of the posterior.
Furthermore, greater uncertainty in the system leads to a wider Gaussian estimate, accentuating the effect of nonlinearity and reducing the accuracy. Therefore, it is essential to keep the uncertainty in the estimate low to achieve good performance.

In the next two paragraphs, a walkthrough the workflow of the EKF implementation for this thesis work is presented.

3.2.2 Prediction Phase

The state vector is defined as:

$$x_t = [a_x, a_y, a_z, v_x, v_y, v_z, dp_x, dp_y, dp_z]$$

The choice of having the difference in position instead of the absolute position is found in the need to keep the propagating error as low as possible. In the prediction phase, the motion model is applied at $\sim 100Hz$ to the data measured by the IMU, in order to predict the mean of the Gaussian representing the state of the system. Since the IMU data is very noisy, a filter must be applied to it in order to have a smooth and reliable prediction: in this work, the linear acceleration and angular velocity are considered as a mobile mean of the last k samples measured by the IMU. The higher k, the smoother the prediction, but the more dynamic of the system is lost. Therefore, the choice of this constant is determined by the frequency at which the IMU measures and some consideration on the application of the algorithm: the faster the dynamics of the system (i.e. a drone), the lower that number must be. Taking into account a sampling frequency of the IMU of $\sim 500Hz$ and the fact that the application is deployed on a rover-like robot, the choice was to set $k = 10$. The chosen motion model is a constant acceleration approximation:

$$\begin{cases} \bar{a}_t = a_{t-1} \\ \bar{v}_t = v_{t-1} + (a_{t-1} - g)\Delta t \\ \bar{p}_t = p_{t-1} + v_{t-1}\Delta t + \frac{1}{2}(a_{t-1} - g)\Delta t^2 \end{cases}$$

where g, that is the acceleration of gravity, and every other kinematic variable, are expressed in the world frame, while Δt is the difference in time between the first and the tenth IMU measurement.
The output of the motion model is the mean of the gaussian estimating the state of the system. In this phase, the covariance of that gaussian and the Jacobians relative to the motion model are updated as well.
Every time the prediction is carried out, the Δp found is added to the previous, in such a way that, when the update is performed, the cumulative information of movements estimated since the last update, is available.
The Jacobians of the motion model with respect to the state and with respect to the command, useful for the EKF algorithm, are computed each step, since the

variables that make them up vary each step. Their form is the following:

$$G_t = \begin{bmatrix} RdR & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & RdR & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & RdR & 0 & 0 & 0 & 0 & 0 & 0 \\ RdR\Delta t & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & RdR\Delta t & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & RdR\Delta t & 0 & 0 & 1 & 0 & 0 & 0 \\ RdR\frac{1}{2}\Delta t^2 & 0 & 0 & \Delta t & 0 & 0 & 0 & 0 & 0 \\ 0 & RdR\frac{1}{2}\Delta t^2 & 0 & 0 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 0 & RdR\frac{1}{2}\Delta t^2 & 0 & 0 & \Delta t & 0 & 0 & 0 \end{bmatrix}$$

$$V_t = \begin{bmatrix} RdR & 0 & 0 & 0 & 0 & 0 \\ 0 & RdR & 0 & 0 & 0 & 0 \\ 0 & 0 & RdR & 0 & 0 & 0 \\ RdR\Delta t & 0 & 0 & 0 & 0 & 0 \\ 0 & RdR\Delta t & 0 & 0 & 0 & 0 \\ 0 & 0 & RdR\Delta t & 0 & 0 & 0 \\ RdR\frac{1}{2}\Delta t^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & RdR\frac{1}{2}\Delta t^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & RdR\frac{1}{2}\Delta t^2 & 0 & 0 & 0 \end{bmatrix}$$

3.2.3 Update Phase

Every time a new couple of images are captured by the stereo camera, the update is performed. First, the VO algorithm described previously is applied to the new images, in such a way to estimate a certain Δp_{CAM}, to be checked against the Δp_{IMU}. The sensor model is very straightforward, since the output of the VO algorithm is the wanted information itself, the difference in position since the last update.

$$z_t = \Delta p_{VO}$$

Then, the Kalman gain K_t is computed taking into account the Jacobian of the sensor model with respect to the state H_t as well as the covariance of the gaussian of the estimation of the state Σ_t and the noise factor Q_t. Finally, the mean and covariance of the estimation of the state are updated, as a result of the weighted average with the information in output of the prediction and of the VO algorithm, as described by the algorithm in the previous section.
The Jacobian of the sensor model with respect to the state, due to the simplicity of the sensor model itself, is the following constant matrix:

$$H_t = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

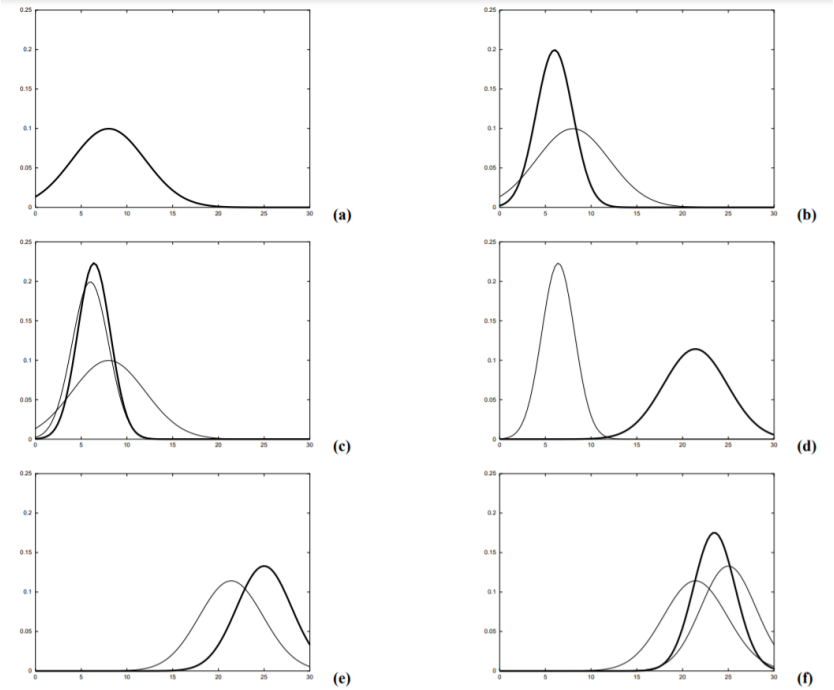


Figure 3.10: Graphical evolution of the gaussian estimating the state

In figure 3.10, the graphical evolution of the Gaussian estimating the state of a system by means of a Kalman filter is shown.

a) Initial belief;

b) Measurement superimposed to the initial belief;

c) New belief after the update, superimposed to the first two Gaussians;

d) New belief after a certain movement, superimposed to the previous belief;

e) New measurement superimposed to the belief

f) There is also the new belief after the update of the last measurement.

Chapter 4

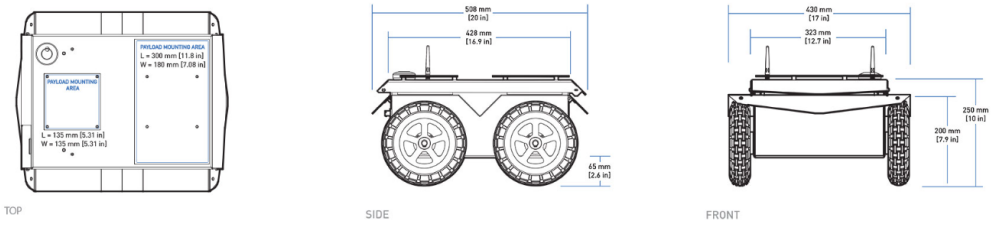
Hardware experimental setup

The hardware setup for deploying the application described in this thesis consists on 3 elements: a robot, a stereo camera and an inertial measurement unit. Those three components are described and referred to in the next subsections.

Figure 4.1: Hardware setup in one of the testing environments

4.1 Jackal

The robot on which the algorithm has been deployed for the indoor tests in the laboratory is the Jackal, created by ClearPath Robotics. It is an unmanned ground vehicle (UGV) built from a sturdy aluminum chassis made with a high torque 4×4 drivetrain for rugged all-terrain operation, capable of moving at a max speed of $2m/s$. Being a compact UGV, it is the best solution for simulating the behavior of a rover, considering the facility where the tests took place.



SIZE AND WEIGHT

EXTERNAL DIMENSIONS	508 x 430 x 250 mm (20 x 17 x 10 in)
INTERNAL DIMENSIONS	250 x 100 x 85 mm (10 x 4 x 3 in)
WEIGHT	17 kg (37 lbs)
MAXIMUM PAYLOAD	20 kg (44 lbs)

SPEED AND PERFORMANCE

MAX SPEED	2.0 m/s (6.6 ft/s)
RUN TIME (BASIC USAGE)	4 hours
USER POWER	5V at 5A, 12V at 10A, 24V at 20A
DRIVERS AND API	ROS 2 Jazzy

Figure 4.2: Technical specifications of the Jackal robot [11]

4.2 Camera

The camera that has been mounted on the Jackal robot is the Intel® RealSense™ Depth Camera D435i. It features 1920×1080 RGB resolution at $30fps$. The following table summarizes all the technical specifications of the camera module.

Metric	**Value**
Baseline	50 mm
Left/Right Imagers Type	Wide
Depth FOV HD (degrees)	H:87±3 / V:58±1 / D:95±3
Depth FOV VGA (degrees)	H:75±3 / V:62±1 / D:89±3
IR Projector	Wide
IR Projector FOV	H:90 / V:63 / D:99
Color Sensor	OV2740
Color Camera FOV	H:69±1 / V:42±1 / D:77±1
IMU	6DoF

Table 4.1: specifications Intel® RealSense™ Depth Camera D435i

Figure 4.3: Intel® RealSense™ Depth Camera D435i

4.3 Inertial Measurement Unit

The Inertial Measurement Unit (IMU) exploited for the deployment of the application is integrated in the camera module described in the previous section. In particular, it is a 6 degrees of freedom IMU (forward/back, up/down, left/right, pitch, yaw, roll) with accelerometer and gyroscope. Table **??** summarizes its specifications.

Parameter	**Properties**
Degrees of Freedom	6
Acceleration Range	±4g
Accelerometer Sample Rate[1]	62.5, 250 (Hz)
Gyroscope Range	±1000 deg/s
Gyroscope Sample Rate[2]	200, 400 (Hz)
Sample Timestamp Accuracy	50 usec

Table 4.2: IMU specifications.

[1] The sample rate may differ from the absolute specified sample rate by ±5%. It is advised to rely on the sample timestamp.

[2] The sample rate may differ from the absolute specified sample rate by ±0.3%.

Chapter 5

Results and discussion

In this section, the results of the tests performed are shown and explained. Following the development process of the system, the tests for the VO algorithm alone are displayed first, followed by the tests of the entire VIO pipeline.
The test campaign for the VIO algorithm was designed in order to check its capabilities in both indoor and outdoor environments and to asses its limits. Therefore, the system was tested in three settings: in the laboratory, in a vineyard and in a special facility resembling a non-terrestrial environment. In each of those three situations, a different aspect of the setting tests a specific limit of the algorithm.
The structure of the result showout, that will be common of all subsections, is the following:

- scatter plots showing the estimated path in parallel with the ground truth path;
- a box plot showing the evolution in time of the relative pose error;
- table summarizing all the relevant metrics for the quantitative evaluation.

Even before that, the metrics adopted to evaluate the quality of the results and the method applied to align the trajectories are defined and explained.

5.1 Metrics and performances evaluation

In order to quantitatively assess the accuracy of the estimated trajectory with respect to the ground truth, several standard metrics are considered. Let $\hat{\mathbf{p}}_i \in \mathbb{R}^d$ denote the estimated position at index i, and $\mathbf{p}_i \in \mathbb{R}^d$ the corresponding ground truth position. The metrics employed are the following:

Absolute Trajectory Error (ATE). The Absolute Trajectory Error measures the global deviation between the estimated trajectory and the ground truth. The per-frame error is defined as:

$$e_i = \left\| (\hat{\mathbf{p}}_i) - \mathbf{p}_i \right\|_2.$$

The overall ATE is summarized using different statistics of the error sequence $\{e_i\}_{i=1}^{N}$, such as the mean, minimum, maximum, and in particular the Root Mean Square Error (RMSE):

$$\mathrm{ATE}_{\mathrm{RMSE}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} e_i^2}.$$

ATE reflects how well the estimated trajectory matches the global shape and scale of the ground truth trajectory.

Relative Pose Error (RPE). The Relative Pose Error evaluates the local consistency of the trajectory over a fixed interval Δk. For each segment, the displacement in the estimated and ground truth trajectories is compared:

$$\Delta\hat{\mathbf{p}}_i = \hat{\mathbf{p}}_{i+\Delta k} - \hat{\mathbf{p}}_i, \qquad \Delta\mathbf{p}_i = \mathbf{p}_{i+\Delta k} - \mathbf{p}_i.$$

The relative translational error is then defined as:

$$r_i = \frac{\left| \|\Delta\hat{\mathbf{p}}_i\|_2 - \|\Delta\mathbf{p}_i\|_2 \right|}{\|\Delta\mathbf{p}_i\|_2}.$$

The RPE is reported through the same statistics as the ATE. This metric highlights local drift and scale errors.

Hausdorff Distance. The Hausdorff distance quantifies the maximum discrepancy between two sets of points. For two trajectories $X = \{\mathbf{x}_i\}$ and $Y = \{\mathbf{y}_j\}$, the directed Hausdorff distance is:

$$d(X, Y) = \max_{\mathbf{x}\in X} \min_{\mathbf{y}\in Y} \|\mathbf{x} - \mathbf{y}\|_2,$$

This represents the largest spatial deviation between the two trajectories.

Path Length. The path length corresponds to the total traveled distance, computed as the cumulative sum of segment lengths:

$$L(\mathbf{p}_{1:N}) = \sum_{i=1}^{N-1} \left\| \mathbf{p}_{i+1} - \mathbf{p}_i \right\|_2.$$

Comparing the estimated and ground truth path lengths allows evaluating systematic scale errors.

Final Drift. The final drift is the Euclidean distance between the last aligned estimated pose and the corresponding ground truth pose:

$$D_{\text{final}} = \left\| \mathcal{S}(\hat{\mathbf{p}}_N) - \mathbf{p}_N \right\|_2.$$

It can also be expressed as a percentage of the ground truth path length:

$$D_{\%} = \frac{D_{\text{final}}}{L(\mathbf{p}_{1:N})} \times 100\%.$$

This metric expresses how far the estimated trajectory drifts from the ground truth at the end of the sequence.

5.2 Trajectory alignment

Before the evaluation, the two trajectories are aligned using a similarity transformation (rotation and translation), fixing a coincident starting point. The method applied is the Kabsch–Umeyama algorithm: it finds the optimal translation, rotation and scaling by minimizing the root-mean-square deviation (RMSD) of the point pairs. The algorithm is as follows:
The input is the two vectors of points describing the ground truth and the estimated path:

$$gt = \Big(\mathbf{a}_1, \ldots, \mathbf{a}_n\Big), \qquad est = \Big(\mathbf{b}_1, \ldots, \mathbf{b}_n\Big)$$

Let the centroids be:

$$\boldsymbol{\mu}_{gt} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{a}_i, \qquad \boldsymbol{\mu}_{est} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{b}_i$$

Define the variance of gt:

$$\sigma_{gt}^2 = \frac{1}{n}\sum_{i=1}^{n} \left\| \mathbf{a}_i - \boldsymbol{\mu}_{gt} \right\|^2$$

Compute the covariance matrix:

$$H = \frac{1}{n}\sum_{i=1}^{n} \left(\mathbf{a}_i - \boldsymbol{\mu}_{gt}\right)^\top \left(\mathbf{b}_i - \boldsymbol{\mu}_{est}\right)$$

Then perform singular value decomposition:

$$H = UDV^\top$$

To detect and prevent reflection, compute:

$$d = \text{sgn}\Big(\det(U)\ \det(V^\top)\Big)$$
$$S = \text{diag}\Big(1, \ldots, 1, d\Big)$$

Finally, the optimal rotation matrix R and scale factor c are given by:

$$R = USV^\top, \qquad c = \frac{\sigma_{gt}^2}{\text{tr}(DS)}$$

Then each aligned point $\mathbf{b}'_i$ is:

$$\mathbf{b}'_i = \boldsymbol{\mu}_{gt} + c\,R\Big(\mathbf{b}_i - \boldsymbol{\mu}_{est}\Big)$$

Alternatively, defining the translation vector

$$\mathbf{t} = \boldsymbol{\mu}_{gt} - c\,R\,\boldsymbol{\mu}_{est}$$

we have the equivalent form:

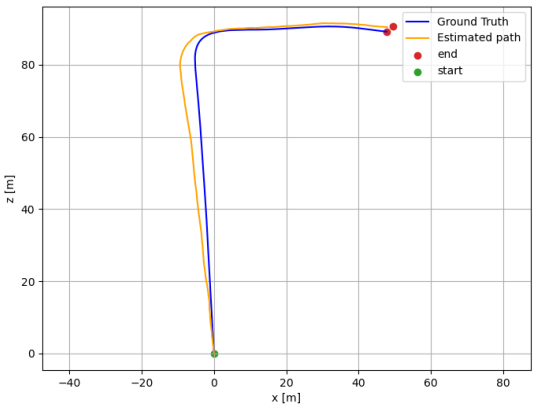
$$\mathbf{b}'_i = \mathbf{t} + c\,R\,\mathbf{b}_i$$

5.3 VO algorithm tests

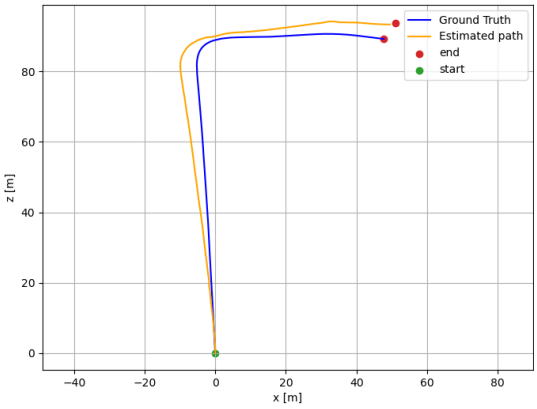
The VO algorithm is the first component of the system that has been developed chronologically. This has first been tested on the KITTI vision benchmark suite [12], that is the most used database for visual odometry tests, later on the data recorded in the three real life scenarios, the very same that have been used to test the whole algorithm.
For the purpose of validation, an alternative pipeline for Visual Odometry (in the following: Traditional VO) has been implemented. This algorithm differs from the Deep VO algorithm under test for the feature detection and matching phases, that are not entrusted to Deep Neural Networks. In particular a FAST keypoint detector [13], a BRIEF descriptor [14] and a FLANN matcher are employed.

5.3.1 VO algorithm tests on KITTI database

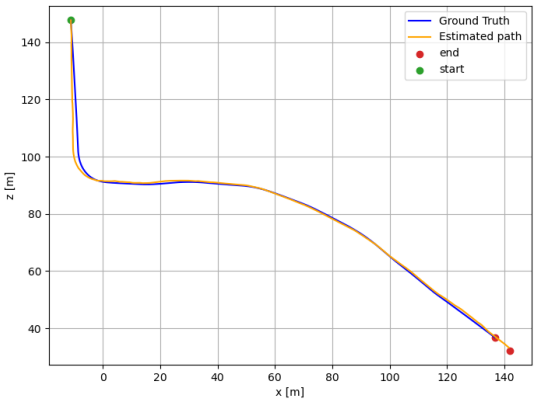
The system endeavored for recording the data of the KITTI database is composed of two high-resolution color and gray-scale video cameras, while the ground truth is provided by a Velodyne laser scanner and a GPS localization system. The videos were recorded by driving in the city of Karlsruhe in rural areas or in highways and up to 15 cars and 30 pedestrians can be seen in the images.
The purpose of the operation was to create datasets for tasks such as optical flow, visual odometry, 3D object detection, and 3D tracking: in the context of this thesis, the 2012 high definition stereo camera dataset for visual odometry is used.
In the following, the results of the simulations are shown.



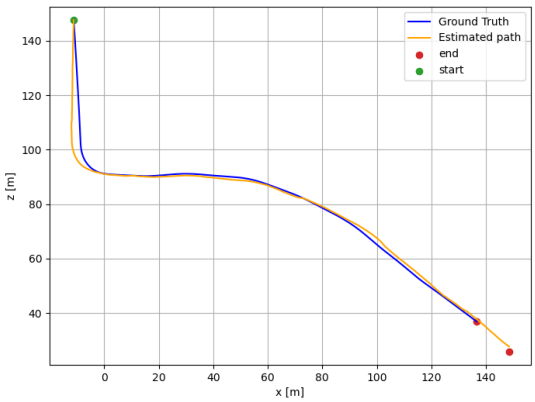
(a) Deep VO trajectory (test 1)



(b) Traditional VO trajectory (test 1)



(c) Deep VO trajectory (test 2)

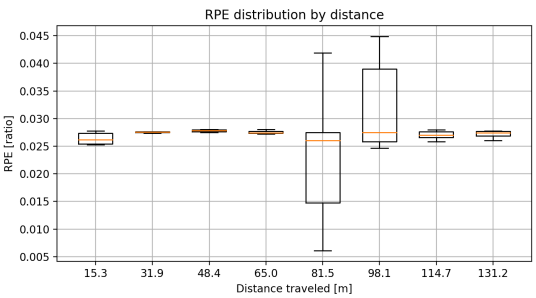


(d) Traditional VO trajectory (test 2)

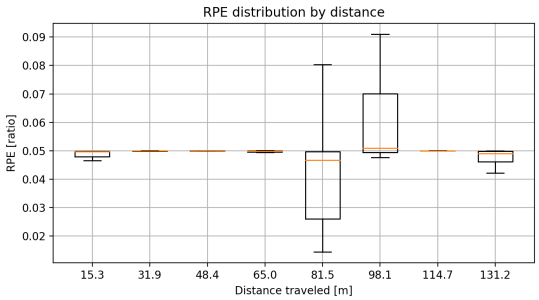
Figure 5.1: Trajectory estimation of the two kitti tests, comparing traditional and deep algorithms

As qualitatively clear from the scatter plots in figure 5.1, and quantitatively clear from the metrics enclosed in table 5.1, the Deep VO version of the algorithm outperforms the traditional version in the matter of estimating the path correctly. In particular the path in output from the Deep VO algorithm results closer to the ground truth with respect to the other one.
The results are very good in terms of numerical errors, since the quality of the data set is very high and the environment where the photos have been taken is full of recognizable visual features.

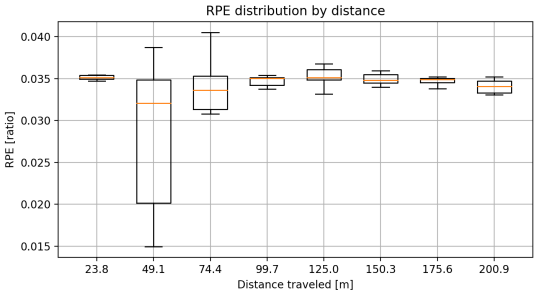
The box plots in figure 5.2 show the trend of the RPE over the distance traveled. In both tests and for both the algorithms, it is clear that the major errors coincide with the 90° curve.



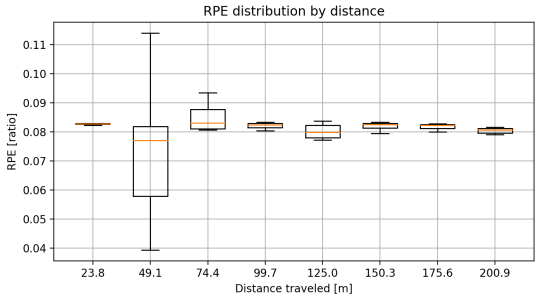
(a) Deep VO RPE box plot (test 1)



(b) Traditional VO RPE box plot (test 1)



(c) Deep VO RPE box plot (test 2)



(d) Traditional VO RPE box plot (test 2)

Figure 5.2: RPE box plot for the two kitti tests, comparing traditional and deep algorithms

The price to pay for the extra accuracy from the neural networks is the speed of execution of the algorithm: while the traditional VO method runs at $12fps$, the Deep version runs at $4fps$. These metrics have to be considered measured on a Intel (R) Core (TM) i7-1065G7 CPU @1.2GHz and its integrated GPU, an Intel (R) Iris Plus Graphics G7. Considering the results obtained, this section of the algorithm is a solid foundation to build the entire VIO pipeline.

As mentioned above, the numerical errors are low, in relation with the length of the path, that is a determining characteristic for evaluating the absolute value of the metrics. For what regards the deep version of the algorithm in particular, the scale factor is close to 1.0 and the path length is only 0.027% longer in the first test with respect to ground truth and 0.034% longer in the second test with respect to the same reference.

Table 5.1: Trajectory metrics report

	Path 1		**Path 2**	
Metric	**Traditional VO**	**Deep VO**	**Traditional VO**	**Deep VO**
ATE [m]				
RMSE	3.358	1.767	9.254	3.966
Mean	3.120	1.631	8.289	3.546
Max	5.595	3.306	16.247	6.969
Min	1.512	0.687	1.993	1.051
RPE [ratio]				
RMSE	0.050	0.028	0.082	0.034
Mean	0.049	0.027	0.081	0.034
Max	0.091	0.045	0.128	0.046
Min	0.014	0.006	0.039	0.015
Hausdorff distance [m]	5.595	2.529	16.247	6.969
Scale	0.9599	0.9827	0.9174	0.9637
Path Length [m]				
Estimated (aligned)	146.467	143.350	231.251	221.080
Ground Truth	139.492	139.492	213.590	213.590
Difference	+6.975	+3.859	+17.661	+7.490
Final Drift [m]				
Absolute	5.595	2.369	16.247	6.969
Percent of GT length	4.011%	1.698%	7.607%	3.263%

5.4 VIO algorithm tests

After having verified the functionality of the VO algorithm, the EKF was implemented. The parameters regarding the noise of the sensors and the trust in the models have been tuned, based on the results of the experiments in the laboratory, reaching their optimal values.
The estimated path has a shape similar with respect to the shape of the ground truth and overall follows it well, moreover the scale is correct. The RPE box plots also show that the error fro the deep VIO algorithm is limited within acceptable limits, rarely exceeding the value of 0.2. The tables in each of the following subsections are filled with information regarding the metrics that have been considered for the quantitative evaluation of the algorithm. For the algorithm under test, the absolute trajectory error and the final path length difference result in average respectively 2% and 6% of the ground truth path length.
The test results are deeply explained in the following subsections.

5.4.1 Test in the PIC4SeR laboratory

Many tests have been performed in the PIC4SeR facility before starting to test in outdoor settings. In this subsection the most relevant is shown.
This indoor setting has very peculiar objects so the keypoints are robust and generally easy to match.

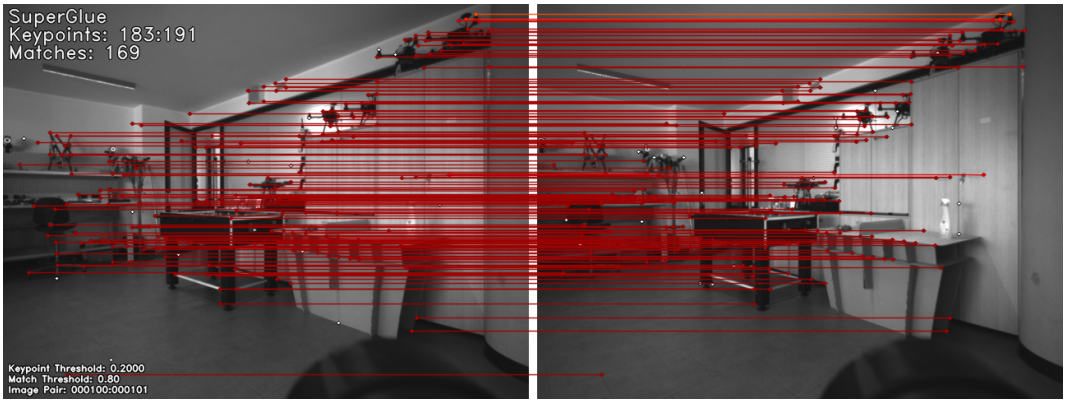
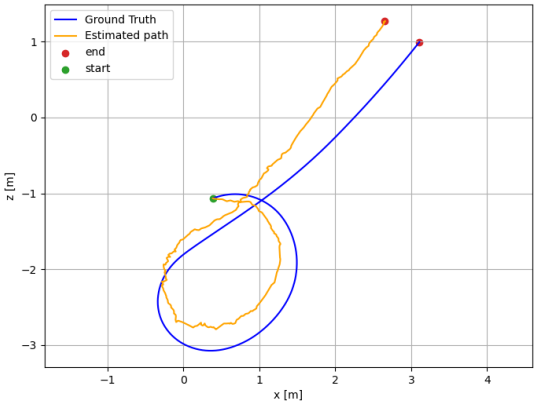


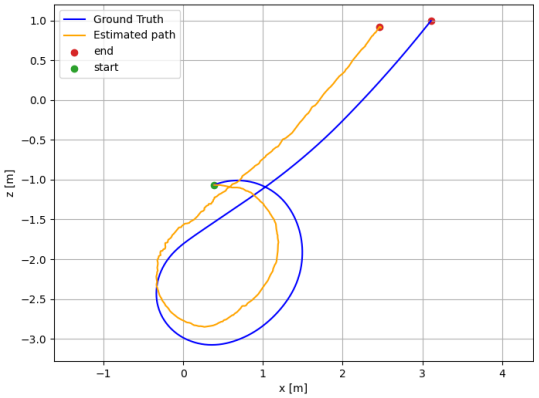
Figure 5.3: Keypoints Matches in the PIC4SeR environment

The trajectory under test consists of a 270° turn followed by a straight: this kind of curve allow to test the ability of the algorithms to track the robot position over a long-lasting rotational movement. It is an effective way to test the algorithm, near the limits related to path complexity. Despite that, it is a quite short path, compared to the following tests, being $9.4m$ long.

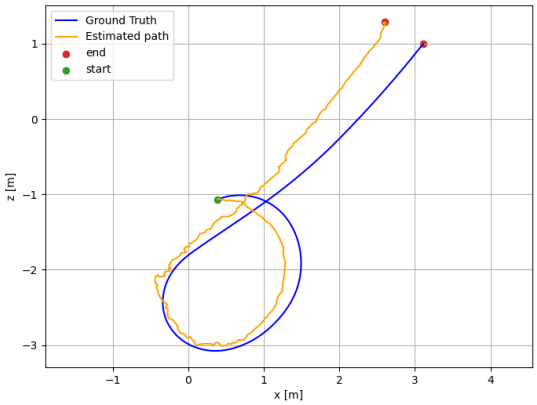
As can be seen from the plots, deep versions of the algorithms produces a smoother trajectory with respect to the traditional ones. Moreover, fusing inertial and visual data permits to better recover the scale of the trajectory, as can be evaluated qualitatively referring to the curve in the path.



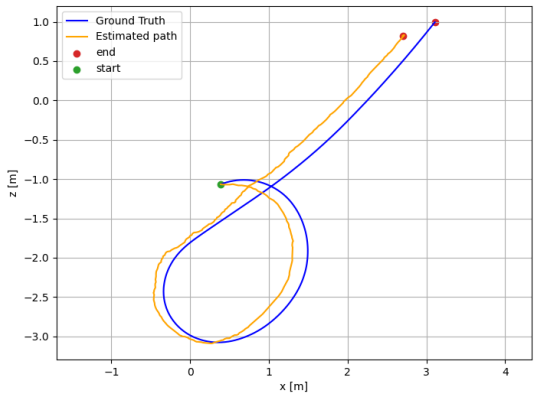
(a) Traditional VO



(b) Deep VO

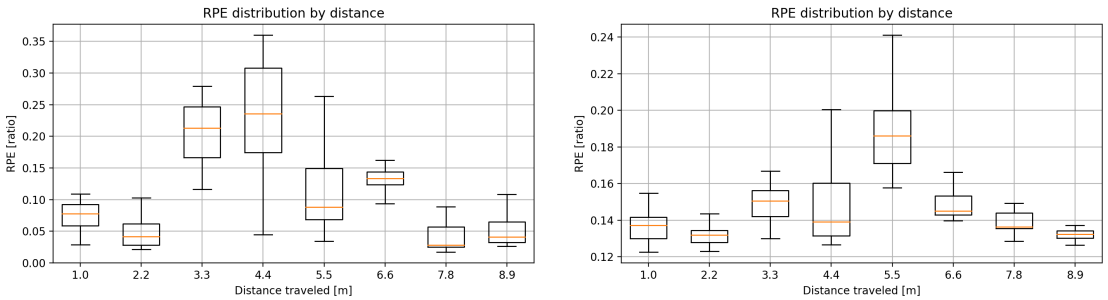


(c) Traditional VIO



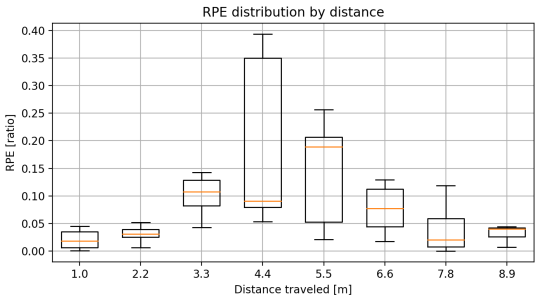
(d) Deep VIO

Figure 5.4: Trajectory estimation of the 4 algorithms, compared

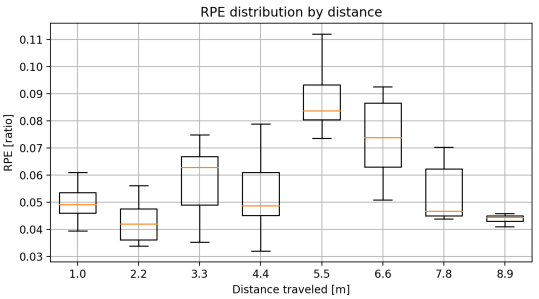


(a) Traditional VO

(b) Deep VO



(c) Traditional VIO



(d) Deep VIO

Figure 5.5: RPE box plots of the 4 algorithms, compared

The box plots demonstrate that the most critical segment of the trajectory is indeed the curve, as the Relative Pose Error (RPE) is higher compared to the initial and final sections of the path.
Quantitatively, the absolute error values are substantially lower than those observed in the KITTI dataset, primarily due to the significantly shorter trajectory length. In this scenario, certain metrics, such as path length and absolute drift, favor the traditional algorithm versions, although both absolute and relative errors are overall improved in the deep learning-based variants.

Table 5.2: Trajectory metrics report

Metric	Traditional VO	Deep VO	Traditional VIO	Deep VIO
ATE [m]				
RMSE	0.545	0.466	0.423	0.252
Mean	0.508	0.421	0.383	0.233
Max	1.015	0.804	0.794	0.449
Min	0.177	0.051	0.084	0.062
RPE [ratio]				
RMSE	0.161	0.154	0.107	0.061
Mean	0.148	0.151	0.082	0.058
Max	0.335	0.226	0.273	0.116
Min	0.047	0.080	0.000	0.010
Hausdorff distance [m]	0.479	0.518	0.386	0.333
Scale	1.050	1.097	0.991	1.029
Path Length [m]				
Estimated (aligned)	18.704	17.633	20.330	19.525
Ground Truth	20.436	20.436	20.436	20.436
Difference	-1.732	-2.803	-0.106	-0.911
Final Drift [m]				
Absolute	0.193	0.582	0.103	0.333
Percent of GT length	0.942%	2.847%	0.502%	1.630%

5.4.2 Test in the vineyard

The first tests in an outdoor setting have been run in a vineyard near Turin. The peculiarity of this setting is the large amount of features that are however similar among each other, being them all lees and trees: this might deceive the VO algorithms.

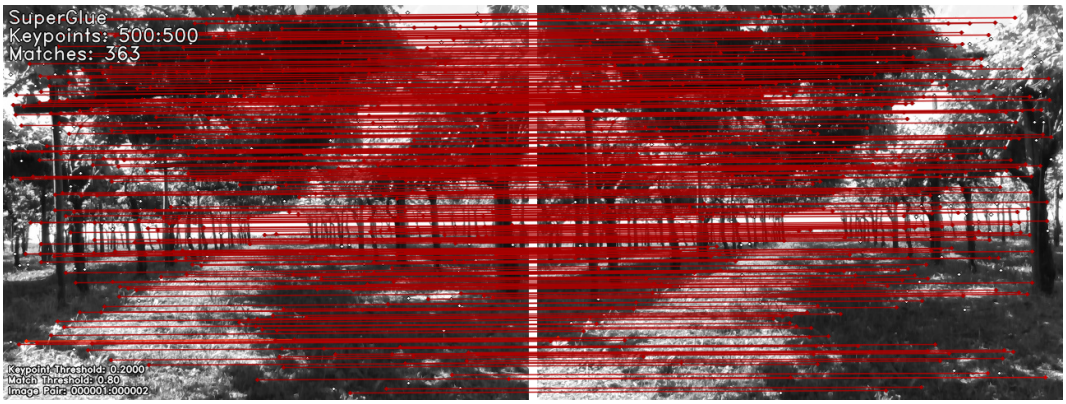
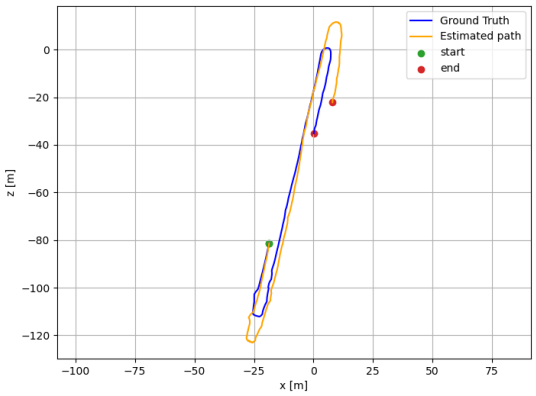


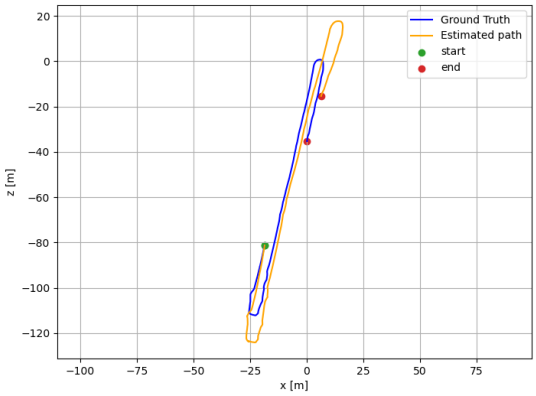
Figure 5.6: Keypoints Matches in the vineyard environment

In this subsection one of tests performed in this environment is shown.
The trajectory consists of a narrow 180° turn, followed by a $100m$ long straight, and lastly another 180° curve. This path has been chosen for two reasons. First, the narrow curves allow for the test of the ability of the algorithm to evaluate how much the robot turns around, being a wide curve easier than a narrow one. The second reason is that the long corridor allows the computation of the drift error accumulated over a long-lasting period of time.

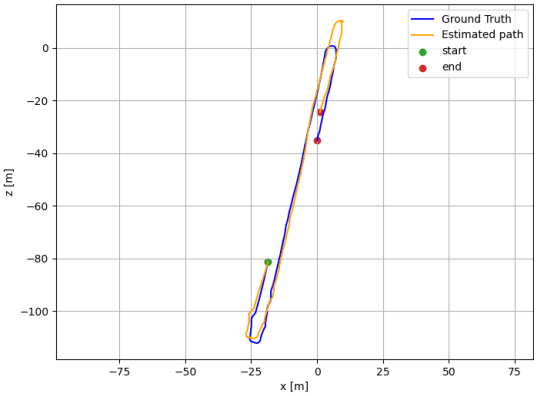
Since the trajectory is very long, it is difficult to appreciate the local differences between the four algorithms. What is evident instead, is that the VO-only variants the drift error accumulates faster, since the first curve is already further than what should be. Moreover, the traditional VO algorithm doesn't evaluate well the second turn, being its exit not parallel with the ground truth.



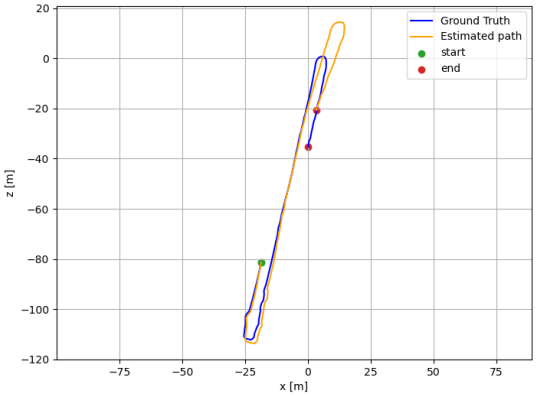
(a) Traditional VO



(b) Deep VO



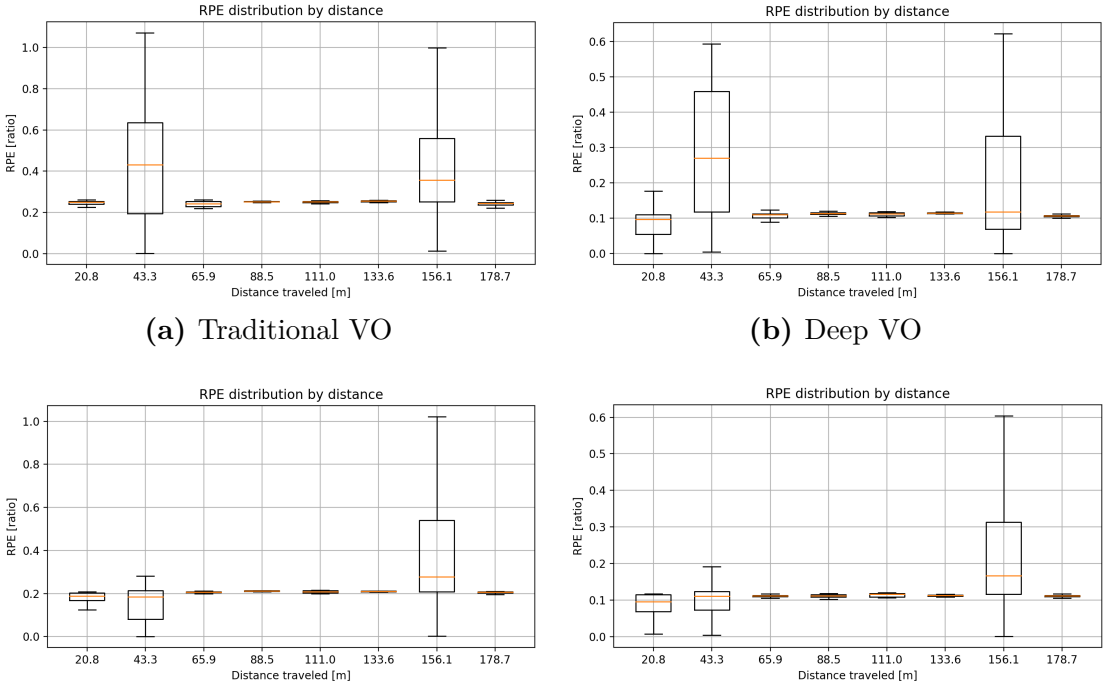
(c) Traditional VIO



(d) Deep VIO

Figure 5.7: Trajectory estimation of the 4 algorithms, compared

The box plots underlines that the relative pose error is higher in correspondence of the turns, with the peak errors being two in the VO-only variants, while the versions that leverage also inertial data presents only the second relative pose error peak.



(a) Traditional VO **(b)** Deep VO

(c) Traditional VIO **(d)** Deep VIO

Figure 5.8: RPE box plots of the 4 algorithms, compared

In metrics table, it iss clear that the absolute errors, such as ATE, path length and final drift, are larger than in the previous tests, since the long corridor has caused for the drift error to accumulate. The relative pose error instead is low, since that metric is not affected by drift accumulated over time.

Table 5.3: Trajectory metrics report

Metric	Traditional VO	Deep VO	Traditional VIO	Deep VIO
ATE [m]				
RMSE	14.011	13.160	8.129	7.686
Mean	10.647	10.242	6.314	6.135
Max	30.039	28.985	17.186	15.406
Min	0.663	1.423	0.616	0.101
RPE [ratio]				
RMSE	0.290	0.361	0.179	0.183
Mean	0.229	0.299	0.131	0.144
Max	1.575	1.578	1.109	0.623
Min	0.000	0.001	0.000	0.000
Hausdorff distance [m]	24.577	19.959	14.031	11.172
Scale	0.777	0.789	0.860	0.869
Path Length [m]				
Estimated (aligned)	227.585	231.473	209.742	204.855
Ground Truth	189.964	189.964	189.964	189.964
Difference	+37.621	+41.509	+19.778	+14.892
Final Drift [m]				
Absolute	23.901	23.427	13.593	14.036
Percent of GT length	12.582%	12.332%	7.156%	7.389%

5.4.3 Test in the Rover eXploration facilit

The last tests have been performed in the Rover eXploration facilitY (ROXY) located at Thales Alenia Space, in Turin. The peculiarity of this setting is the absence of clear visual features, being the camera pointed downwords and being the soil bare. This is the most complex situation for a VO algorithm, since the precision and variety of the correspondences are crucial. The peculiarity of the terrain serves the purpose of resembling a non-terrestrial ground.

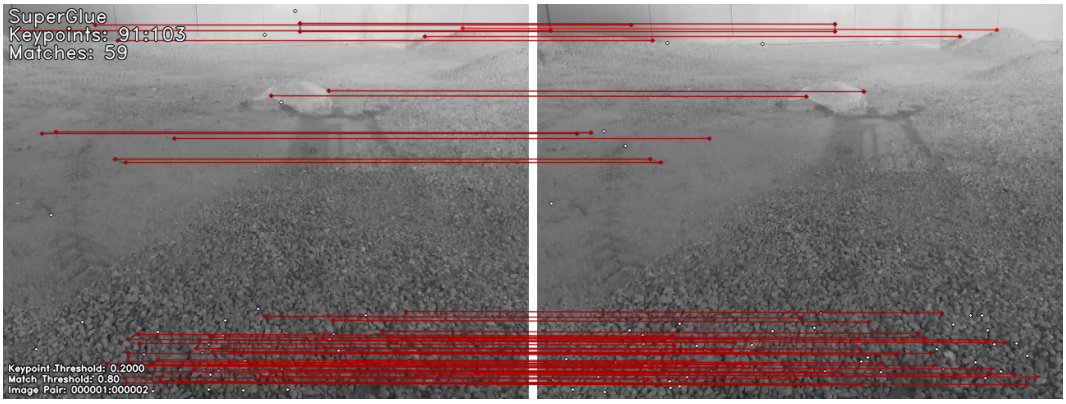
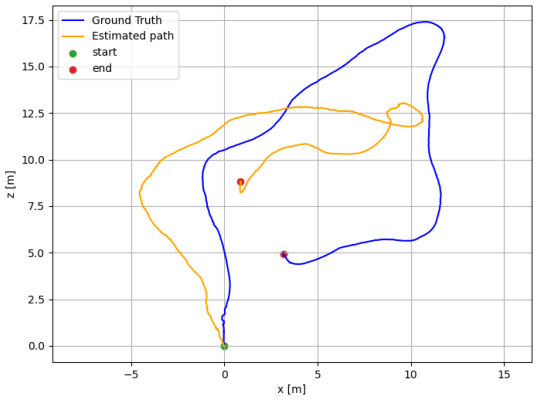


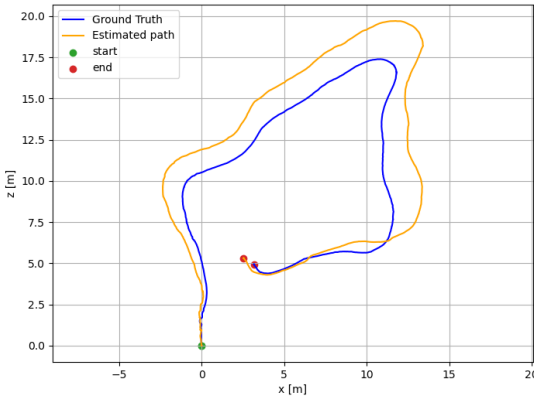
Figure 5.9: Keypoints Matches in a non-terrestrial envirnment

In this subsection one of tests performed in this facility is shown.
The trajectory seems simple from a 2D perspective, even tough the terrains presents bumps and hills that might trick the algorithm.

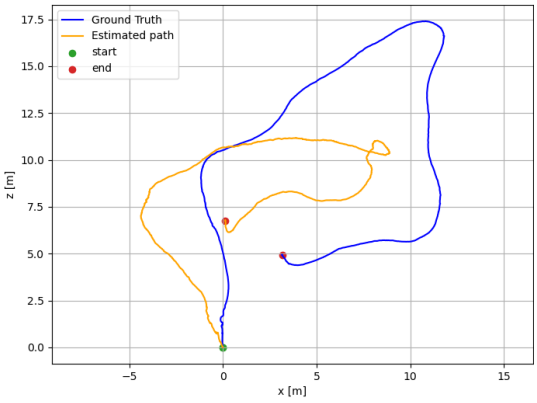
As can be seen from the plots of the estimated trajectories, the traditional algorithms fail to track the robot movements, in a certain region. This is due to the lack of clear geometrical features, that the algorithm is looking for, due to the camera being very close to the terrain in that section. This problem is solved by the neural networks in deep variant the algorithm that are able to stay on track. Moreover it can be qualitatively seen that the deep VIO has a tracking with respect to the deep VO algorithm alone, being closer to the ground truth.



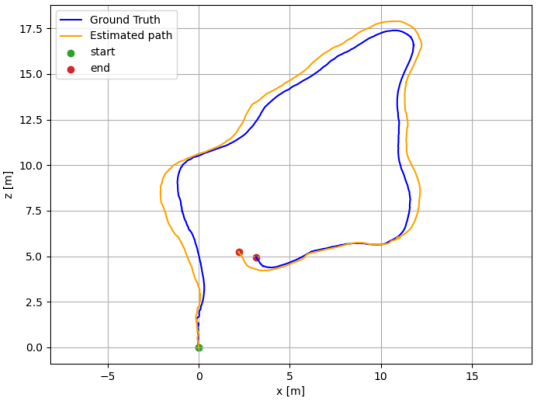
(a) Traditional VO



(b) Deep VO



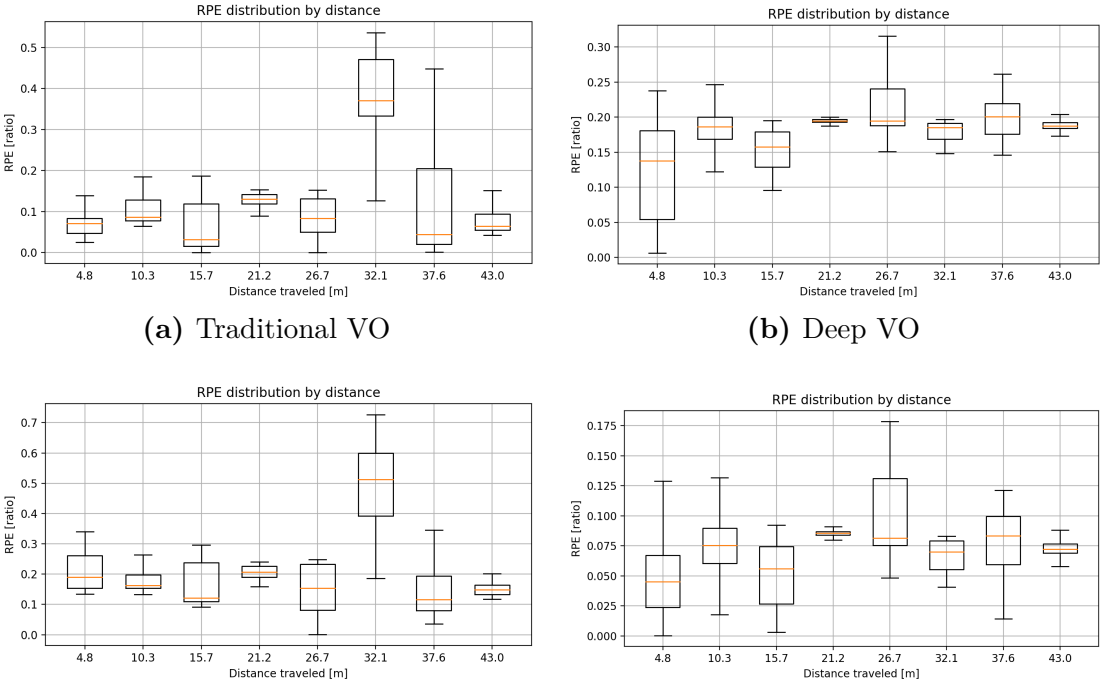
(c) Traditional VIO



(d) Deep VIO

Figure 5.10: Trajectory estimation of the 4 algorithms, compared

The scale on the RPE axis on the box plots show the difference between the errors in among traditional and deep variants, due to the fact that the tracking is being lost in path section mentioned above. Moreover, the Deep VIO algorithm outperforms the VO-only one, in terms of average and maximum RPE.



(a) Traditional VO **(b)** Deep VO

(c) Traditional VIO **(d)** Deep VIO

Figure 5.11: RPE box plots of the 4 algorithms, compared

Since the qualitative analysis enlighted the failure of the traditional algorithms, their metrics loose value. For example, the path length predicted by the traditional VIO algorithm is closer to the real one, with respect to the one predicted by the deep VO algorithm: this is irrelevant, since the trajectory itself is wrong.

Table 5.4: Trajectory metrics report

Metric	Traditional VO	Deep VO	Traditional VIO	Deep VIO
ATE [m]				
RMSE	3.193	2.120	3.296	0.851
Mean	2.759	1.926	2.823	0.776
Max	5.539	3.353	6.892	1.432
Min	0.257	0.379	0.201	0.072
RPE [m]				
RMSE	0.177	0.185	0.254	0.078
Mean	0.128	0.179	0.215	0.072
Max	0.536	0.333	0.726	0.179
Min	0.000	0.006	0.001	0.000
Hausdorff distance [m]	5.521	3.273	6.844	1.430
Scale	1.008	0.856	1.181	0.942
Path Length [m]				
Estimated (aligned)	42.397	54.359	38.703	49.366
Ground Truth	45.773	45.773	45.773	45.773
Difference	-3.376	8.586	-7.070	3.592
Final Drift [m]				
Absolute	4.138	0.772	2.376	0.383
Percent of GT length	9.040%	1.687%	5.191%	0.837%

Chapter 6

Conclusions

This thesis presented the design, implementation, and comprehensive evaluation of a hybrid deep learning-based visual-inertial odometry system. By leveraging the computational power of convolutional neural networks for feature detection and matching and an Extended Kalman Filter for sensor fusion, the proposed system finds a trade off among the strengths and weaknesses of both worlds.
The experimental results obtained across multiple real-world scenarios demonstrate that the deep learning components significantly enhance pose estimation accuracy and trajectory smoothness compared to traditional methods. Moreover, in challenging conditions where visual information alone is insufficient, inertial data proves to be useful to improve robustness. From the results, the integration of the neural networks turns out to be worth it, despite a computational overhead leading to slower executions with respect to classical pipelines.
Among the algorithms tested, the deep visual-inertial odometry variant can achieve the assigned task with superior accuracy. This is evident from the systematic evaluation with metrics such as Absolute Trajectory Error, Relative Pose Error, Hausdorff distance and final drift. Moreover, the deep models recover scale more reliably, reducing drift accumulation effectively.
Future research directions could focus on optimizing computational efficiency through model compression or hardware acceleration, and exploring other ways to further exploit inertial data.
Overall, this work contributes to advancing autonomous navigation capabilities by demonstrating how modern deep learning techniques can be synergistically combined with established geometrical methods to obtain state-of-the-art performances in visual-inertial odometry.

Bibliography

[1] LAKMAL SENEVIRATNE YUSRA ALKENDI and YAHYA ZWEIRI. «State of the Art in Vision-Based Localization Techniques for Autonomous Navigation Systems». In: *IEEE ACCESS* (2021), pp. 76847–76874 (cit. on p. 3).

[2] HONGKAI WEN SEN WANG RONALD CLARK and NIKI TRIGONI. «DeepVO: Towards end-to-end visual odometry with deep Recurrent Convolutional Neural Networks». In: *IEEE EXPLORE* (2017) (cit. on p. 4).

[3] J. M. M. Montiel Raul Mur-Artal and Juan D. Tardos. «ORB-SLAM: a Versatile and Accurate Monocular SLAM System». In: *IEE TRANSACTIONS ON ROBOTICS* (2015) (cit. on p. 7).

[4] Tong Qin, Peiliang Li, and Shaojie Shen. «VINS-Mono». In: *IEEE Transactions on Robotics* 34.4 (Aug. 2018), pp. 1004–1020. ISSN: 1941-0468. DOI: 10.1109/tro.2018.2853729. URL: http://dx.doi.org/10.1109/TRO.2018.2853729 (cit. on p. 12).

[5] B. D. Lucas and T. Kanade. «An iterative image registration technique with an application to stereo vision». In: *Proc. Int. Joint Conf. Artif. Intell* (1981) (cit. on p. 12).

[6] Arman Neyestani, Francesco Picariello, Amin Basiri, Pasquale Daponte, and Luca De Vito. «Survey and Research Challenges in Monocular Visual Odometry». In: 2023, pp. 107–112 (cit. on p. 17).

[7] Davide Scaramuzza and Friedrich Fraundorfer. «Visual Odometry [Tutorial]». In: *IEEE Robotics Automation Magazine* (2011) (cit. on pp. 17–19, 24).

[8] Tomasz Malisiewicz Daniel DeTone and Andrew Rabinovich. «SuperPoint: Self-Supervised Interest Point Detection and Description». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2018) (cit. on pp. 17, 19, 20).

[9] Tomasz Malisiewicz Paul-Edouard Sarlin Daniel DeTone and Andrew Rabinovich. «SuperGlue: Learning Feature Matching With Graph Neural Networks». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020) (cit. on pp. 17, 22).

[10] Wikipedia contributors. *Kalman filter — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/wiki/Kalman_filter`. [Accessed: 25 August 2025]. 2025 (cit. on p. 25).

[11] ClearPath Robotics. *technical specifications of the Jackal robot*. `https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/`. [Accessed: 15 September 2025] (cit. on p. 33).

[12] 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 (cit. on p. 39).

[13] Miroslav Trajković and Mark Hedley. «Fast corner detection». In: *Image and Vision Computing* 16.2 (1998), pp. 75–87 (cit. on p. 39).

[14] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. «BRIEF: Binary Robust Independent Elementary Features». In: *Computer Vision – ECCV 2010*. Ed. by Kostas Daniilidis, Petros Maragos, and Nikos Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 778–792 (cit. on p. 39).

Acknowledgements

Ringrazio innanzitutto il mio relatore, il Prof. Marcello Chiaberge, per l'opportunità di lavorare a questo progetto di tesi all'interno del suo laboratorio. Ringrazio i correlatori ufficiali e non, Mauro Martini, Marco Ambrosio e il prof. Stefano Primatesta, per avermi supportato durante tutto il percorso, per la disponibilità e pazienza nel prendere parte alle riunioni che si sono rivelate necessarie nel corso di questi mesi al PIC. Vi ringrazio inoltre per aver acconsentito a farmi portare la Jetson dall'altra parte d'Italia, così vicino al mare. Sono contento di aver scelto questo percorso di tesi, che possa indirizzarmi nel corso della mia futura carriera?

Ai miei genitori, senza di voi gli obiettivi che ho raggiunto e le esperienze che ho potuto vivere in questi due anni, sarebbero state solo fantasia. Grazie per aver creduto in me tanto da fare i tanti sacrifici che mi hanno permesso di avere questa opportunità: ve ne sono immensamente grato. Grazie per le nostre chiamate serali sono servite tanto a voi quanto a me: siete riusciti a farmi sentire a tavola con voi anche a 1000km di distanza. Stare lontano da voi mi ha fatto rendere ancor più conto del valore del tempo passato insieme. E come sempre grazie per sopportarmi, visto che finite sempre per essere la mia valvola di sfogo.

A mio fratello, sono contento di come il nostro rapporto stia crescendo con noi. Sembra ieri che ti urlavo addosso perchè ti mettevi sempre in bolla a Super Mario Bros. Grazie per l'interesse genuino che dimostri e grazie per i consigli che mi dai, il tuo parere è importante. Sai che puoi contare su di me e io so che posso contare su di te: questo è tutto. Continua così che vai lontano.

A cicia, la mia fortuna. Ogni pagina di questo percorso porta con se un po' del tuo sorriso, delle tue attenzioni e della tua pazienza. Con la tua sensibilità senza eguali e le tue parole sempre ponderate riesci a capirmi anche quando non ti parlo, riesci ad accarezzare le corde giuste, lasciando che ogni mio dubbio trovi pace in un tuo abbraccio. Se sono arrivato fin qui, è anche merito tuo: sei stata la mia motivazione, il mio equilibrio e la mia serenità nei momenti più difficili, senza

mai pretendere nulla in cambio. Grazie per avermi insegnato a mandare in fumo ciò che non posso controllare, ad abbandonarmi alle onde, a vivere con più serenità e leggerezza. Questa vita ci ha sorriso e lo sai, perchè ho al mio fianco la persona con cui voglio tagliare ogni traguardo, la persona con cui, anche quando poi saremo stanchi, troveremo il modo per navigare nel buio.

A Marco e Elisa, mi avete accolto tra di voi sin dal primo giorno, quando ancora non sapevamo quanto duro sarebbe stato quel primo semestre... Grazie per avermi aiutato nell'inserimento in quella che per me era una nuova realtà, in particolare con il corso accelerato di VHDL, senza il quale oggi sarei ancora alle prese con il VGA controller. A Bat, che ha reso un'esperienza di studio da fuorisede, un corso avanzato di trekking, alpinismo e mountain bike. Grazie per avermi fatto assaporare quella che è la tua quotidianità, senza mai chiedre nulla in cambio, salvo qualche nodino. Dopo 12 corsi e altrettanti lab insieme, posso dire che noi 4 siamo stati una squadra vincente, a cui gli altri chiedevano quando non riuscivano in qualcosa: sono fiero di noi. Ma soprattuto siete gli amici che cercavo qui, sono contento di aver condiviso questi due anni con voi.

Alle vecchie conoscenze de "Il Meridione", siamo stati scaraventati tutti nella stessa situazione bene o male contemporaneamente e ci siamo fatti forza a vicenda. Abbiamo legato molto dalla triennale e sono davvero contento per questo, spero che i nostri legami non si sciolgano con il futuro incerto che ci aspsetta. Siete stati fondamentali per portare un po' di sano terronismo in questo contesto nordico (x es gli unici con cui cenare alle 9:30 come le persone normali!). Abbiamo finanziato e visto fallire Burgo's, speriamo che almeno Fratelli Pummaro' regga dai.

Ad Andrea e Pier, abbiamo condiviso questa prima esperienza di vita da soli, ma non da soli. Tra le minacce della signora di sotto, le panzerottate che si respirano ancora nell'aria, la cucina andata a fuoco, la casa completamente allagata, le innumerevoli cose che abbiamo rotto, siamo ancora qui: la reputo una vittoria.

A Ciccio, il nostro rapporto si è stretto molto da quando mi sono trasferito e per questo sono un sacco contento. Grazie per avermi portato ai miei primi spettacoli di cabaret e per avermi accompagnato al cinema anche a vedere film che ti saresti risparmiato altrimenti. E in generale grazie per la tua disponibilità nei miei confronti, lo apprezzo davvero. Che ci creda o no, tra i giorni più divertenti che ricordo a Torino, c'è quello in cui abbiamo fatto il trasloco!

Agli amici di sempre, che siete venuti fin qui a Torino da svariate città d'Italia per esserci in questo giorno per me memorabile: siete degli persone da custodire. Grazie per esservi sempre fatti trovare presenti ogni volta che tornavo a casa, facendomi

sentire come se non fossi mai partito. La distanza ha solo reso più chiaro quanto siano rari i legami che resistono al tempo e ai chilometri. Non servono grandi gesti: la semplicità con cui ci ritroviamo vale più di tutto.
Non molti sanno cosa si provi a prenotare un viaggio di sola andata, lasciando alle spalle casa, familiari e amici. Ho imparato più che mai in questi due anni che i rapporti interpersonali vanno coltivati con impegno, che non si può dare nulla per scontato. Per questo motivo, ci tengo particolarmente a ringraziare Matteo, Bobby, Claudia, Nora e Simona che, grazie anche alla loro esperienza simile alla mia, hanno saputo cogliere questo aspetto.

Infine, ci tengo specialmente a ringraziare la mia nonnina, che durante questa laurea magistrale si è ritrovata ad avere in mano delle redini molto importanti e ha saputo ammorbidire quello che sarebbe altrimenti stato un ostacolo molto duro. Sentire alle 19 in punto la tua risata è uno dei momenti preferiti della mia giornata, non posso farne a meno. Parlando di cose di cui non posso fare a meno, il tuo ragù e il tuo minestrone, accuratamente centellinati nel corso della mia permanenza a Torino, sono stati fondamentali per la mia sopravvivenza lontano dal cibo di casa! A parte gli scherzi nonna, grazie per la leggerezza che mi trasmetti con le tue parole curiose e innocenti, senza pretese ma con tante aspettative, che cercherò sempre di rispettare.