PhD-FSTM-2022-098
The Faculty of Science, Technology and Medicine

DISSERTATION

Defence held on 29/09/2022 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE
by

CLAUDIO CIMARELLI
Born on 22 August 1991 in Rome, Italy

PERCEPTION FOR SURVEILLANCE: LEARNING SELF-LOCALISATION AND INTRUDERS DETECTION FROM MONOCULAR IMAGES OF AN AERIAL ROBOT IN OUTDOOR URBAN ENVIRONMENTS

UNIVERSITY OF LUXEMBOURG

FACULTY OF SCIENCE, TECHNOLOGY AND MEDICINE

AUTOMATION AND ROBOTICS RESEARCH GROUP

DOCTORAL THESIS

Perception for Surveillance: Learning Self-Localisation and Intruders Detection from Monocular Images of an Aerial Robot in Outdoor Urban Environments

Thesis Submitted in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy in Computer Science

Author:
Claudio CIMARELLI

Supervisors:
Prof. Dr.-Ing. Holger VOOS
Prof. Dr. Miguel Angel OLIVARES MENDEZ
Prof. Dr. Radu STATE

To the love of my life.

“If I have seen further, it is by standing on the shoulders of giants. ”
Sir. Isaac Newton

Acknowledgments

First, I express my deep gratitude to Prof. Dr.-Ing. Holger Voos for having welcomed me to the Automation and Robotics Research Group and let me pursue my research interests throughout the project, following his vision. I am honoured to have him as my thesis supervisor for his kindness and profound knowledge.

I want to thank Prof. Dr Radu State for his support and the valuable feedback during our meetings. Also, a special thank you to Prof. Dr Miguel A. Olivares-Mendez, with whom I shared many experimental sessions, a man whom I admire, and I wish him great success with his new space group adventure.

I wish to mention the LuxConnect company and thank them for having started this project and funding the initial phases of the PhD.

To Dr Dario Cazzato, thank you for being a research mentor and a great friend. Thanks to Dr Marco Leo, Dr Cosimo Distante, and Dr Pierluigi Carcagnì for the significant collaborations that improved my skills as a computer vision student. Thank you, Dr Jose Luis Sanchez-Lopes, for your guidance in research and as a motivating group leader, for after-work beers, and for the excellent time together. And to Dr Hriday Bavle, whose knowledgeable presence in the last year greatly impacted my research.

I wish to thank all the colleagues, especially Manuel, Philippe, Paul, Raja, Amin, Serket, Anush, Gary, and Eduardo, whom I met since I started this journey in the Automation and Robotics Research Group, for which I am grateful to consider as friends. Also, a special mention to Manuel for his compelling charisma and positive attitude. To Philippe, with whom I shared many discussions and beneficial exchanges on our similar research topic and had an essential role in shaping the final thesis. To Paul, a great passionate engineer that inspires me constantly to improve and never stop learning. Then, a thank you to all the newcomers in the group, and particularly my super office mates Alireza and Meisam, Shaheer, Ahmed, Ali, and Eduardo, and the new PostDocs Dr Davide Menzio, Dr Hamed Habibi, Dr Hamed Rahimi, wishing them good look with their future research in the university.

For having led me to this robotics path and giving trust to me, I would like to mention Dr Patrice Care. Thank you for being a great leader of the Luxembourg United team, seeking creative and innovative research, and inspiring others.

Finally, I thank my family, my mother and dad, for their support and vicinity, even at a distance. And my sister, for the presence in the joyful moments.

For my wife, the most important person in my life whom I love and thank immensely for her patience, gentleness, and strength, I dedicate all the efforts that led me here.

Abstract

Unmanned aerial vehicles (UAVs), more commonly named drones, are one of the most versatile robotic platforms for their high mobility and low-cost design. Therefore, they have been applied to numerous civil applications. These robots generally can complete autonomous or semi-autonomous missions by undertaking complex calculations on their autopilot system based on the sensors' observations to control their attitude and speed and to plan and track a trajectory for navigating in a possibly unknown environment without human intervention. However, to enable higher degrees of autonomy, the perception system is paramount for extracting valuable knowledge that allows interaction with the external world.

Therefore, this thesis aims to solve the core perception challenges of an autonomous surveillance application carried out by an aerial robot in an outdoor urban environment. We address a simplified use case of patrolling mission to monitor a confined area around buildings that is supposedly under access restriction. Hence, we identify the main research questions involved in this application context. On the one hand, the drone has to locate itself in a controlled navigation environment, keep track of its pose while flying, and understand the geometrical structure of the 3D scene around it. On the other hand, the surveillance mission entails detecting and localising people in the monitored area. Consequently, we develop numerous methodologies to address these challenging questions. Furthermore, constraining the UAV's sensor array to a monocular RGB camera, we approach the raised problems with algorithms in the computer vision field.

First, we train a neural network with an unsupervised learning paradigm to predict the drone ego-motion and the geometrical scene structure. Hence, we introduce a novel algorithm that integrates a model-free epipolar method to adjust online the rotational drift of the trajectory estimated by the trained pose network. Second, we employ an efficient Convolutional Neural Network (CNN) architecture to regress the UAV global metric pose directly from a single colour image.

Moreover, we investigate how dynamic objects in the camera field of view affect the localisation performance of such an approach. Following, we discuss the implementation of an object detection network and derive the equations to find the 3D position of the detected people in a reconstructed environment. Next, we describe the theory behind structure-from-motion and use it to recreate a 3D model of a dataset recorded with a drone at the University of Luxembourg's Belval campus.

Ultimately, we perform multiple experiments to validate and evaluate our proposed algorithms with other state-of-the-art methodologies. Results show the superiority of our methods in different metrics. Also, in our analysis, we determine the limitations and highlight the benefits of the adopted strategies compared to other approaches. Finally, the introduced dataset provides an additional tool for benchmarking perception algorithms and future application developments.

Contents

Acknowledgments i

Abstract iii

List of Figures vi

List of Tables viii

Acronyms x

1 Introduction 1
- 1.1 Perception for Aerial Surveillance 3
- 1.2 Visual Localisation and Mapping 6
- 1.3 From Traditional to Deep SLAM 9
- 1.4 Deep Learning for Object Detection 14
- 1.5 Objective, Scope, and Contributions 16
- 1.6 Thesis Structure 21

2 Deep Learning Background 22
- 2.1 Neural Networks Mathematics 25
- 2.2 Convolutional Neural Network 38

3 Unsupervised Ego-Motion and Depth Estimation 43
- 3.1 Literature Review 46
- 3.2 Unsupervised Learning of Depth and Ego-Motion 49
- 3.3 Geometrical Loss and Rotation Adjustment 53
- 3.4 Networks Architectures 55
- 3.5 Experiments 56
- 3.6 Discussion 63
- 3.7 Summary 70

4 Supervised Global Pose Estimation 71
4.1 Literature Review . . . 73
4.2 End-to-End Pose Regression . . . 76
4.3 Experiments and Results . . . 80
4.4 Pose Estimation in Dynamic Scenes . . . 86
4.5 Summary . . . 99

5 Intrusion Detection in Outdoor Environments 100
5.1 Related Work . . . 102
5.2 2D Object Detection . . . 103
5.3 3D Object Localisation . . . 109
5.4 Environment 3D Reconstruction . . . 112
5.5 Experiments and Results . . . 118
5.6 Summary . . . 136

6 Conclusion 137

References 173

List of Figures

1.1 Autonomous System Overview 4
1.2 Factor Graph graphical model 8
1.3 Example of commercial drones 17

2.1 Abstraction of a neuron 25
2.2 A model of an artificial neuron 26
2.3 Example of a multi-layer perceptron (MLP) 27
2.4 Logistic sigmoid and hyperbolic tangent functions 28
2.5 ReLU functions and their variations 30
2.6 Dropout effect visualisation 36
2.7 Example of a CNN architecture 38
2.8 Convolution operation types compared 39
2.9 A numerical example of the avg. and max pooling operations 40

3.1 Rotational Adjusted Unsupervised Visual Odometry (RAUM-VO) block diagram. 45
3.2 Diagram of RAUM-VO training. 57
3.3 KITTI train trajectories 62
3.4 KITTI test trajectories 63

4.1 Architecture of Mobile-PoseNet 78
4.2 7-Scenes sample images 80
4.3 Cambridge Landmarks sample images 80
4.4 Cumulative distribution of the localisation error 83
4.5 Cumulative distribution of the localisation error 84
4.6 King's College sample pictures from the Cambridge Landmarks dataset . . 88
4.7 Pre-processing applied to a sample image of the King's College sequence . . 89
4.8 Cumulative distribution plot of the translation and rotation error 90
4.9 Box plots of the translation and rotation errors 92

4.10 Translation and rotation errors grouped by occluded pixels percentage . . . 93
4.11 Histogram of the percentage of the masked pixels 94
4.12 Saliency maps with a person in the foreground 97
4.13 Saliency maps with vehicles in the background 98

5.1 Feature Pyramid Network illustration 104
5.2 EfficientDet Architecture 105
5.3 SmoothL_1 loss plot 108
5.4 Representation of the 3D object localisation 110
5.5 Factor graph representing the SfM problem 113
5.6 COLMAP sparse reconstruction pipeline 114
5.7 Plots of the adjacency matrix of the scene graph 116
5.8 AU-AIR sparse reconstruction 119
5.9 AU-AIR 3D reconstruction stats 120
5.10 AU-AIR dense reconstruction 121
5.11 3D locations of detected people in the dataset 124
5.12 Plot of the learning rate decay and pose train loss 126
5.13 Plot of the median rotation error during training 126
5.14 Plot of the median translation error during training 127
5.15 Cumulative distribution of the localisation error 128
5.16 Belval dataset GPS coordinates 129
5.17 Belval 3D reconstruction statistics 130
5.18 Belval 3D sparse reconstruction 130
5.19 Belval train trajectories 132
5.20 Belval test trajectories 133
5.21 Example of estimated depth for Belval train sequences 134
5.22 Example of estimated depth for Belval test sequences 135

List of Tables

3.1 Odometry quantitative evaluation on KITTI odometry seq. 00–10. 61
3.2 Network prediction results separately mixed with ground-truth data. 65
3.3 frame-to-frame (F2F) solver initialization 66
3.4 perspective-n-point (PnP) vs pose network 67
3.5 Scale alignment. 69

4.1 Median localisation errors on the 7-Scenes and Cambridge Landmarks datasets 85
4.2 Number of objects detected per each category 88
4.3 Median and mean errors w/ and w/o object masking 90

5.1 AU-AIR object detection results 123
5.2 Global Localisation Results on AU-AIR 127
5.3 Odometry quantitative evaluation. 131

Acronyms

AI artificial intelligence.

AP average precision.

AR average recall.

ATE Absolute Trajectory Error.

BA bundle adjustment.

BCE Binary Cross Entropy.

BoVW bag-of-visual-words.

CL continual learning.

CNN Convolutional Neural Network.

DL Deep Learning.

DNN Deep Neural Network.

DoF degrees of freedom.

DPM Deformable Parts Model.

EASA European Union Aviation Safety Agency.

ELU Exponential Linear Unit.

F2F frame-to-frame.

FAST Features from Accelerated Segment Test.

FLOPS floating point operations.

FoV	field-of-view.
FPN	Feature Pyramid Network.
GCS	geographic coordinate system.
GN	Gauss–Newton.
GNN	Graph Neural Network.
GNSS	Global Navigation Satellite System.
GPS	Global Positioning System.
GPU	graphics processing unit.
GRIC	geometric robust information criterion.
HOG	Histogram of Gradients.
ICP	iterative closest point.
IMU	Inertial Measurement Unit.
IoU	Intersection over Union.
IR	image retrieval.
LIDAR	Light Detection and Ranging.
LM	Levenberg–Marquardt.
madds	multiply additions.
MAP	Maximum a Posteriori.
ML	machine learning.
MLP	Multi-Layer Perceptron.
NAS	Neural Architecture Search.
NeRF	neural radiance fields.
NIST	National Institute of Standards and Technology.

NLP natural language processing.

NMS Non-Maximum Suppression.

OF optical flow.

ORB ORB (Oriented FAST and Rotated BRIEF).

PGO pose graph optimization.

PnP perspective-n-point.

RADAR Radio Detection and Ranging.

RANSAC random sample consensus.

RAUM-VO Rotational Adjusted Unsupervised Visual Odometry.

ReLU Rectified Linear Unit.

RGB red green blue.

RGB-D red green blue and depth.

RMSE Root Mean Square Error.

RoI region of interest.

RPE Relative Pose Error.

RPN Region Proposal Network.

SDK software development kit.

SfM structure from motion.

SGD Stochastic Gradient Descent.

SIFT Scale-Invariant Feature Transform.

SLAM simultaneous localization and mapping.

SotA state-of-the-art.

SSD Single Shot Detector.

SSIM	Structural Similarity Metric.
STN	Spatial Transformer Network.
SVM	Support Vector Machines.
UAV	Unmanned Aerial Vehicle.
VBL	visual based localization.
VIO	visual inertial odometry.
VJ	Viola and Jones.
VO	visual odometry.
VPR	visual place recognition.
V-SLAM	visual simultaneous localization and mapping.
YOLO	You-Only-Look-Once.

1 Introduction

The recent decade's significant leap forward in robotics has enabled numerous civil applications [250] to adopt advanced technology, which has become widely available through normal commercial channels. Among the technological platforms open to the general public, aerial robots are the most versatile due to their high mobility and low-cost design. Feron and Johnson [253, p.1011] attribute the term "aerial robot" to Robert Michelson. He used it to describe intelligent flying machines capable of accomplishing complex missions with various degrees of autonomy. We use the more popular concept of Unmanned Aerial Vehicle (UAV), which includes multiple types of drones that can be teleoperated by a human pilot or execute a planned flight assisted by the onboard controller and sensor system. We generally distinguish UAVs that are "lighter-than-air", such as blimps, from the more familiar "heavier-than-air", *e.g.*, wing type and multi-rotors [171]. The multiple design options give each kind of UAV unique qualities to fit various task requirements. For this reason, as confirmed by Liew *et al.*. [171], aerial robotics is an incredibly active field of research that has grown steadily over the years, constantly sets new challenges, and opens new prospects for application in diversified areas.

Surveillance is one of the scenarios which is appealing to automate the patrolling task carried out by humans. The proper robotic platform can alleviate the repetitiveness and reduce the time for each patrol, primarily when the area under surveillance covers large outdoor spaces. Hence, for this use case, the high mobility and operation range of UAVs grants human security personnel additional support from an elevated observation point. Furthermore, multi-rotor drones are particularly easy to manoeuvre and have the ability to hover, which becomes particularly helpful in maintaining a fixed position above an observation target. An autopilot system [37], comprising an onboard processor and several sensors for observing the robot's states, such as an Inertial Measurement Unit (IMU), a Global Positioning System (GPS) antenna, and a barometer or an altimeter, can exploit these flight capabilities through autonomous control software algorithms [41]. Quoting Pachter and Chandler [209], autonomous control implies that an "high degree of automation is applied in a very unstructured environment" to solve complex optimisation problems. Adapting this concept to UAVs autonomous navigation, the robot plans a trajectory in a possibly unknown environment without human intervention and incurs uncertainties of different types, *e.g.*, dynamics, measurement noise, and cooperative or non-cooperative multi-agent systems.

Furthermore, the National Institute of Standards and Technology (NIST) [122] defines the levels of autonomy based on mission complexity, environment complexity, and human operator independence. To the extreme autonomy spectrum, "all decisions are made on-board based on sensors observations adapting to operational and environmental changes" [73]. The complexity of the surveillance patrolling missions lies in monitoring a specified area by moving freely and detecting security breaches. The UAV needs to perceive the environment to avoid obstacles (primarily static if flying above human height), localise itself, and assist or even substitute the human guard in its duty.

Perception entails understanding the information in the signals gathered by the sensors from the surrounding environment and extracting valuable knowledge that allows the interaction of an intelligent agent with the external world. With control and planning, perception is one of the core components of an autonomous system [276]. Taking inspiration from Pendleton *et al.*. [212], Figure 1.1 describes a high-level abstraction of the software architecture that places these three core elements at the centre while restricting the consideration of hardware and external interactions at the limit. Among other notable representations of an autonomous system, it is worth mentioning the Aerostack architecture [237], which shares the essential concepts with the proposed diagram. However, it emphasises the hierarchical structure of decision-making modules that compose, as a whole, the autonomous behaviour of an intelligent agent with a multi-layered architecture and aims to guide the software implementation. Instead, we intend to identify the principal research areas regarding autonomous control, fundamental to realising an intelligent system, to direct our attention to a single topic.

1.1 Perception for Aerial Surveillance

The present thesis focuses on investigating solutions for the perceptual level of an autonomous UAV that faces the challenges of a surveillance mission. Hence, the two sub-topics of perception, localisation, *i.e.*, self-perception and environmental perception, represent the spheres of competence of the following discussion. On the one hand, localisation deals with ego-motion tracking and global pose estimation with respect to a global coordinate system defined at the

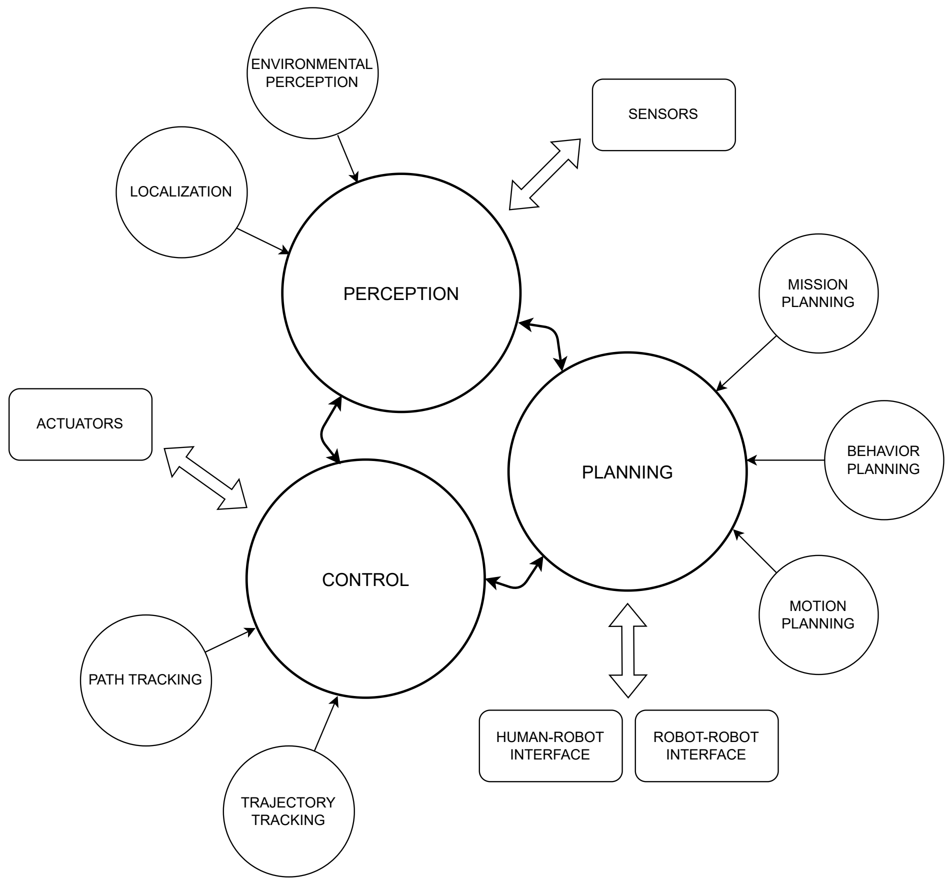


Figure 1.1: Autonomous System Overview

The figure shows an abstraction of the main components of an autonomous system software architecture.

centre of a map [282, p.159]. When these problems relate to a 3D environment, the pose describes the 6 degrees of freedom (DoF) coordinate transformation from the centre of the map to the robot view. It contains a 3D position and a 3D orientation. On the other hand, environmental perception encompasses a broad and diversified set of problems ranging from depth estimation [200, 246] to object detection [321] and semantic segmentation [95] that develop a comprehensive understanding of the surrounding elements. Together, localisation and environmental perception are the premises for a situational awareness system that also involves comprehending the meaning of objects, reasoning about cause and effect implications, and projecting the current states of the perceived elements into the future [13, 75].

Similarly to how humans use words to convey meanings and establish relations among these to form complex concepts, robots also need a mechanism to map environmental elements to conceptual entities. Thus, natural language processing (NLP) [116] cannot be excluded from a complete situational awareness system as it is the foundation for reasoning. NLP regards the translation between human and machine languages to establish an active channel for communication with robots [6, 146]. Hence, the representation of Figure 1.1 assimilates it with the sphere of perception and among human-robot interface technologies. However, we limit the comprehension capability of an autonomous agent to answering the questions: *"Where am I located?"*, *"What is the structure of the surrounding environment?"*, and *"Which objects lie in my field of perception and where?"*, for which NLP is not required.

Considering the use case of surveillance, which constitutes the application context of this thesis, we decline these questions into a practical task definition. Hence, we define the aerial surveillance task as detecting people in a restricted access zone around a building or, more generally, in a confined urban area. While there may be an interest in adding people's face identification [319] and human behaviour recognition [218], our primary concern is enabling autonomous navigation in this confined area under surveillance and, subordinate to this, detecting people. Autonomous navigation requires that all of the components of an autonomous system, *i.e.*, perception, planning, and control, are developed [188]. However, the previous questions push our efforts towards perception, *i.e.*, localisation and environmental perception, which gives us the instruments to answer them. Primarily, localisation is fundamental as it allows the UAV to move around and contributes to discovering the position of the people that are possibly detected. Secondly, a representation of the environment, *e.g.*, a map, is needed in conjunction with localisation to plan the trajectory and place detected humans inside or outside the restricted access area. Finally, once the previous aspects are addressed, human detection can be included among the environmental perception capabilities to fulfil the surveillance requirements. Therefore, localisation and mapping are paramount in this context and answer the central questions of perception as defined above.

1.2 Visual Localisation and Mapping

Simultaneous localization and mapping (SLAM) [71] owes its name to the ambition of estimating the ego-motion and understanding the environment structure — to create a map — at the same time [253, p.871]. The modern mathematical formulation of SLAM approaches this problem using a probabilistic framework and through the formalism of factor graphs [26]. Hence, we introduce these concepts partially following the description provided by Cadena *et al.*. [26]'s survey and integrate it with more detailed introductions to factor graphs available in the literature [61, 105, 149, 178]. A graphical model describes a factor graph (see Figure 1.2 for an example) whose visual components correspond to precise mathematical elements of its formulation. Nodes of the graph encode the unknown state variables $\mathcal{X}$ to be estimated. The variables include a discrete set of poses $\{x_1, \ldots, x_n\} \subseteq \mathcal{X}$, which form the robot's trajectory in correspondence with the selected "keyframes", and the locations of the "landmarks" $\{l_1, \ldots, l_m\} \subset \mathcal{X}$, 3D points that compose the representation of the map as perceived by the robot. Notably, state variables may not include landmarks in all formulations, thus receding SLAM to the problem of estimating the robot trajectory alone, called pose graph optimization (PGO) [59]. Edges represent the measurements $Z = \{z_k : k = 1, \ldots, p\}$ between two nodes. Unlike the state variables, measurements z_k are known quantities and can be expressed as a function, named the observation model, of a subset of the states $\mathcal{X}_k$ that are measured, *i.e.*, $z_k = h_k(\mathcal{X}_k) + \varepsilon_k$, where ε_k is random measurement noise. Modelling the noise as corrupted by zero-mean Gaussian noise, we obtain the following conditional density:

$$p(z_k|\mathcal{X}_k) \propto \exp\left\{-\frac{1}{2}\|h_k(\mathcal{X}_k) - z_k\|^2_{\Omega_k}\right\} , \tag{1.1}$$

where Ω_k is the information matrix, the inverse of the covariance. To emphasise $\mathcal{X}$ as the variable to estimate, $p(Z|\mathcal{X})$ can be translated into the likelihood $l(\mathcal{X}; Z)$ of the states $\mathcal{X}$ given the measurements Z. The likelihood may be any function proportional to the conditional density that acts as a parameter [61]. Consequently, likelihoods are not Gaussian in the general case where $h_k(\cdot)$ is non-linear.

Therefore, solving the SLAM problem means finding the Maximum a Posteriori (MAP) estimate of the unknown state variables $\mathcal{X}$ that maximises the posterior density $p(\mathcal{X}|Z)$, also called the belief over $\mathcal{X}$ given the measurements. Following from the Bayes theorem:

$$\begin{aligned}
\mathcal{X}^{\mathrm{MAP}} &= \underset{\mathcal{X}}{\operatorname{argmax}}\, p(\mathcal{X}|Z) && (1.2)\\
&= \underset{\mathcal{X}}{\operatorname{argmax}}\, \frac{p(Z|\mathcal{X})p(\mathcal{X})}{p(Z)} && (1.3)\\
&= \underset{\mathcal{X}}{\operatorname{argmax}}\, l(\mathcal{X};Z)p(\mathcal{X}) \ , && (1.4)
\end{aligned}$$

where $p(\mathcal{X})$ represents the prior knowledge about $\mathcal{X}$. Notably, we exclude the normalisation factor $p(Z)$ because it does not influence the maximisation solution. We define the prior similarly to $p(z_k|\mathcal{X}_k)$:

$$p(\mathcal{X}_0) \propto \exp\left\{-\frac{1}{2}\|h_0(\mathcal{X}_0) - z_0\|^2_{\Omega_0}\right\} \ , \tag{1.5}$$

given an initial measurement function $h_0(\cdot)$ and prior mean z_0.

Dellaert *et al.*. [61] elevate the graph's edges to a particular type of node, the factor. A factor corresponding to a measurement z_i is a function $\varphi_i(\mathcal{X}_i)$ of the adjacent variable nodes $\mathcal{X}_i = \mathcal{N}(\varphi_i)$, where $\mathcal{N}$ denotes the adjacency relationship. From another perspective, factor φ_i connects the subset of states $\mathcal{X}_i$, where $\mathcal{X}_i \subseteq \mathcal{X}$ and $\bigcup_i \mathcal{X}_i = \mathcal{X}$, among which exists a measurement z_i. Factors are more general than conditionals as they include both the likelihood and prior functions and are not restricted to probability densities. Also, assuming independent measurements, they allow to explicitly represent a factorised formulation of the problem in Equation 1.4 as the product of multiple factors:

$$\mathcal{X}^{\mathrm{MAP}} = \underset{\mathcal{X}}{\operatorname{argmax}}\, \varphi(\mathcal{X}) = \underset{\mathcal{X}}{\operatorname{argmax}} \prod_i \varphi_i(\mathcal{X}_i) \ , \tag{1.6}$$

where $\varphi(\mathcal{X}) \propto p(\mathcal{X}|Z)$ is the global function defined by a factor graph. Assuming that all factors are in the same Gaussian form of the likelihood (see Equation 1.1) and the prior (see Equation 1.5) and minimising the negative log of Equation 1.6, the MAP estimate is equal to a non-linear least squares problem:

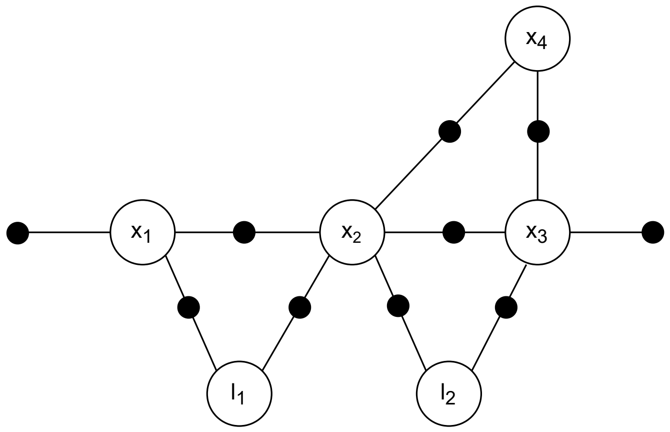


Figure 1.2: Factor Graph graphical model

An example of a factor graph with four pose variables $\{x_1, x_2, x_3, x_4\}$ and two landmarks $\{l_1, l_2\}$. Also, factors are displayed as black nodes. For instance, landmark l_1 is observed from the robot's locations x_1 and x_2. Also, a loop-closure is established between the poses x_2 and x_4. Finally, "unary" factors constrain the poses x_1 and x_3 in the global coordinate system frame of the map, which is implicitly represented in every graph.

$$\mathcal{X}^{\text{MAP}} = \underset{\mathcal{X}}{\text{argmin}} -\log\left(\prod_i \varphi_i(\mathcal{X}_i)\right) = \underset{\mathcal{X}}{\text{argmin}} \sum_i \|h_i(\mathcal{X}_i) - z_i\|^2_{\Omega_i} . \tag{1.7}$$

Modern solvers for the factor graph minimisation problem involve iterative methods, such as Gauss–Newton (GN) or Levenberg–Marquardt (LM), based on successive optimisations of a linear approximation [104]. Currently, many libraries are available for solving the problem efficiently leveraging the sparse structure of the factor graph, among which GTSAM [58], g^2o [150], Ceres [5], and SLAM++ [236] are the most noteworthy.

In a SLAM system, we distinguish two components: the back-end and the front-end. The back-end task is to solve the MAP estimate given some measurements from pre-processed sensor data provided by the front-end. Remarkably, the front-end performs data association between the robot's poses and landmarks to optimise their relative location. Additionally, factors constrain pairs of pose variables using motion models and control inputs or various forms of odometry.

Odometry is the incremental estimation of the robot's pose changes through sensors. Depending on the type of sensor, different techniques have to be applied to extract valuable information, such as landmarks' locations or the robot's ego-motion. Excluding wheel odometers, because they are restricted to ground robots, aerial vehicles commonly rely on IMU-based odometry, Light Detection and Ranging (LIDAR)-based odometry, or visual odometry (VO) [244], which uses RGB or RGB-D cameras. Also, due to the flexibility of factor graphs, fusing multiple sensor data is straightforward after taking time synchronisation into account [285]. For example, the MAP estimation of fused IMU and camera measurements results in a (loosely coupled [245]) visual inertial odometry (VIO) system [40].

Furthermore, when a robot revisits a location, corresponding pose nodes are connected, reducing the error accumulated over the trajectory known as drift. Due to its visual appearance in the graphical model, this factor is called a loop-closure and is easier to recognise using features extracted from visual data, such as bag-of-visual-words (BoVW) [86]. Notably, the growing research topic of visual place recognition (VPR) [89, 186] defines the successful recognition of a place by comparing images with an overlapping field of view, and image retrieval (IR) is the most common declination of this problem [9, 42, 102, 302].

Finally, unary factors, *i.e.*, measurements constraining a single state variable, can represent the knowledge of the current robot's global location, specified relative to the centre of the map. The global localisation is usually possible thanks to a Global Navigation Satellite System (GNSS) such as the GPS [195] but is also achievable with the support of visual based localization (VBL) [216] techniques. Combining VO and VBL methods provides the tools for implementing a visual simultaneous localization and mapping (V-SLAM) front-end [268].

1.3 From Traditional to Deep SLAM

Early V-SLAM methods do not optimise the entire trajectory. Instead, they optimise poses within a moving window covering a few nodes, named fixed-lag smoother [162] or just the last pose. In this, we call filtering approaches those marginalising the previous trajectory in a dense Gaussian prior, such as MonoSLAM [57] or FastSLAM [202]. On the other hand, we refer to the full-smoothing method when the errors and uncertainties are "smoothed" over the history of poses [60]. Strasdat *et al.*. [262] prove the superior smoothing accuracy, showing that it avoids the accumulation of linearisation errors caused by marginalisation. Also, these methods have become increasingly more efficient following breakthrough developments in incremental solvers [245], such as iSAM2 [133], which exploit the topological structure of the factor graph to handle thousands of variables. Therefore, they can track more state variables than filtering, granting more robustness.

Furthermore, traditional V-SLAM approaches are categorised as direct or indirect based on the type of features extracted from images and included in the optimised error function in Equation 1.7. Direct methods such as DTAM [205], LSD–SLAM [77] or DSO [76] include the pixel intensity values or even the actual sensor measurements in a photometric error function. Then, they estimate the motion along with the camera model parameters by minimising a non-linear energy function representing the visual similarity between iteratively aligned images. Depending on the extent of the image portion included in the optimisation, we distinguish dense methods [205, 263] from sparse [76] passing by semi-dense methods [77, 78] that compromise the two options. Instead, indirect methods [57, 93, 142, 203], also known as feature-based, leverage geometrical error functions derived from epipolar geometry theory [107]. The general approach is to generate intermediate feature representations of sparse keypoints detected in correspondence with corners or other salient image points. The second step matches the extracted points and uses minimal sets of five [207] to eight [180], depending on the inclusion of camera parameters in the optimisation problem, for estimating the 2D motion from the essential matrix decomposition [244]. If a 3D map is available, which is the case after triangulating a few image frames, PnP [160] within a robust random sample consensus (RANSAC) fitting scheme [27] obtains better relative pose estimations than 2D motion models. Indirect methods rarely exploit the entire image area for the dense map reconstructions as the feature extraction already implies an additional overhead compared to the direct counterparts. However, techniques extracting dense optical flow (OF), a map of pixel displacements between consecutive images, can detect moving objects, benefiting surveillance tasks [188, 193]. Finally, SVO [84] is one of the few examples of hybrid solutions to VO. Forster *et al.*. [84] compute an initial guess of the motion using direct image alignment and then refine it by minimising the reprojection error of implicitly detected features. In V-SLAM, optimising the non-linear reprojection error function is often treated as a (local or global) bundle adjustment (BA) [286] problem, closely related to MAP formulation. However, BA does not consider other sensor models and cannot be solved incrementally [26].

The advent of Deep Learning (DL) [101, 153] established the predominance of the Deep Neural Network (DNN) [206] as a mathematical model for approximating any arbitrarily complex function, *i.e.*, following from the "universal approximation theorem" [119], through learning and imposing these models as a unified solution to a vast number of problems in artificial intelligence (AI) [7]. Therefore, it is not surprising that SLAM is not an exception to this trend. To date, differently from tasks such as image classification, deep SLAM has not reached the maturity to stand independently from the traditional theory. For example, back-end optimisation can still be a valuable component in a learned system as a post-processing step to further reduce errors or fuse other sensor data.

A prolific line of research focuses on translating traditional SLAM into its Deep Learning version. Many straightforwardly accomplish this transformation by replacing the indirect approach's feature extraction and matching steps with neural networks while keeping the other components unchanged [23, 134, 159, 163, 274, 275]. For example, DF-SLAM [134] substitutes hand-crafted features such as ORB [235] or SIFT [185] by describing patches around FAST [234] corners. Hence, they train the TFeat network [12] using a triplet loss to generate normalised 128-dimensional vectors that identify each input patch distinctly and enable traditional motion tracking and loop closure with BoVW. Similarly, Tang *et al.*. [274] extend their previous work, Geometric Correspondence Network (GCN) [275], to create an alternative to the ORB feature for the popular ORB-SLAM2 [204] that could run more efficiently on a graphics processing unit (GPU). Instead, DXSLAM [163] extracts both local and global features with a typical network architecture HF-NET [239]. This network provides a Superpoint decoder [66] and a NetVLAD layer [9], which are used for relative and global pose estimation. Rather than maintaining a pre-trained Superpoint model, DeTone *et al.*. [65] propose to self-supervise the retraining of the feature extraction network using the output of the same VO. In addition, they add the knowledge of keypoint stability based on the point tracks, the length, and the reprojection error. Notably, the advantages of learned features, which have become more tangible with the progress in the field [72, 198, 208, 228, 309, 310], are the increased robustness to viewpoint changes and adaptation to different lighting conditions caused by variable weather or day-to-night shift. Notwithstanding, the motion tracking failure cases limit the improvements achievable by solely modifying the feature extraction component. Especially with monocular cameras, unstructured visual scenarios, *e.g.*, homogeneous textures and perceptual aliasing, and degenerate motion situations, *e.g.*, pure rotational motion, require complex engineered solutions to recover tracking.

Furthermore, the direct approach has inspired significant advancements toward a complete deep SLAM. CNN-SLAM [277] improves the accuracy of LSD-SLAM by employing a Convolutional Neural Network (CNN) [70, 148, 155] to provide depths and pixel-wise uncertainty for the keyframes, *e.g.*, the images used to create nodes in the PGO that show significant changes in motion. Besides, the authors demonstrate the outstanding advantages of CNN-predicted depths when handling VO with monocular cameras. In particular, the CNN can produce a globally consistent scale, thus reducing the drift. Also, the neural network approach enables continuous pose tracking even in pure rotational motion, which does not allow observing points' distances with the traditional triangulation approach. Recently, Loo *et al.*. [181] applied an analogous strategy to initialise the depth filter of SVO, resulting in better handling of challenging illumination conditions. DTAM is another example of DL adaptions for estimating camera motion and dense depth reconstruction. DeMoN [289] uses an encoder-decoder architecture to simultaneously predict the 6 DoF egomotion, depth maps, optical flow, and normals between two views. DeepTAM [322] separates the tracking from the depth reconstruction with two dedicated networks. The first network estimates slight pose variations that iteratively align the current frame to a reference keyframe. Instead, the mapping network finds the depth of the keyframe using a cost volume composed by accumulating information from multiple images and refines the output by focusing on the initially predicted shapes with a narrow band. DeepV2D [278] approaches the sub-problems of SLAM with two trainable modules and alternates between motion and depth regression. Unlike DeepTAM, it uses an intermediate 2D feature representation to create the cost volume from N images, which are then composed using 3D convolution and pooling operations. CodeSLAM [19] learns via a variational auto-encoder [140] a compact representation of the depth, inspiring future works in pursuing continuously optimisable depth encodings. BA-Net [273] further brings the concept of codes and generates the dense geometrical structure from a linear interpolation of learned basis depth maps. The authors optimise the camera motion and the interpolation weights by minimising the photometric difference of CNN's features in a BA-layer. Notably, they formulate a differentiable LM algorithm to back-propagate the gradients concerning the network features. DeepFactors [53] start from CodeSLAM's idea of depth code optimisation while developing a complete SLAM with loop-closure and global BA. However, this method directly predicts a plausible code with an image encoder network to initialise the MAP problem using a more realistic depth estimation. Finally, DROID-SLAM [280] improves the performance and the generalisation capabilities with a novel "Differentiable Recurrent Optimisation-Inspired Design". The authors compose parts of the RAFT [279] OF architecture with a ConvGRU [12] to produce a revised pixel correspondence field and achieve impressive results. Hence, a Dense Bundle Adjustment Layer (DBA) minimises using GN the Mahalanobis distance of the geometric error between frames connected in a co-visibility graph and obtains a per-pixel precise depth and pose prediction. The main limitations of these methods are the high computational cost and memory demand that mandates modern and expensive GPU equipment.

Continual learning (CL) is getting increasing attention as the paradigm for adapting a machine learning model to new *stimuli* from the real world by including new knowledge while dealing with potential catastrophic forgetting [161]. Concerning deep V-SLAM, Vödisch *et al.*. [294] propose specific performance metrics to measure the adaptation quality (AQ) related to the generalisation of novel out-of-distribution data and the retention quality (RQ), indicating the level of catastrophic loss the system suffers. However, they intend to address a problem broader than what they define as lifelong SLAM. Notably, while lifelong SLAM considers only the adaption to new data from a single dynamic environment, they extend the task to learning multiple scenes with different settings. Instead, Sucar *et al.*. [265] consider CL the reduced problem of continuously acquiring knowledge on a single scene while retaining old information. Hence, they use a replay-based approach that stores past keyframes in a buffer to avoid memory loss of previously visited regions. Also, their method, named iMAP, pioneers the application of neural radiance fields (NeRF) [199] to V-SLAM. In practice, they train a simple neural network composed solely of fully-connected layers, called the Multi-Layer Perceptron (MLP), which maps a pixel ray originating from the camera to colour and density value, hence implicitly representing the scene's appearance and geometry in the network weights. Zhu *et al.*. [327] propose NICE-SLAM, a dense V-SLAM based on a hierarchical neural implicit scene representation that enables it to scale up to larger indoor environments by modelling them with three feature grids of varying space granularity, *e.g.*, coarse, mid and fine levels. Hence, a MLP from a pre-trained ConvONet [213] decodes the space features, and another network learns the colour information similarly to iMAP. Therefore, they integrate a differentiable rendering process inspired by NeRF to predict the geometry and appearance of the scene from a given camera point of view and minimise the loss between the ground truth observations and the predictions. Nevertheless, neural implicit representation methods have not demonstrated the capability of reconstructing outdoor environments larger than a few rooms in an indoor scenario. Nonetheless, the NeRF rendering is a GPU intensive process and requires other techniques to speed up the computations, such as sampling a subset of the image pixels. Also, the training involves ground truth depth maps from an RGB-D camera and excludes the possibility of exploiting datasets consisting of a single camera image stream.

1.4 Deep Learning for Object Detection

The object detection problem relates to the robot perception question posed in section 1.1: *"Which objects lie in my field of perception and where?"*. Under the scope and context of this thesis, this problem has a subordinate role compared to localisation and mapping. Nevertheless, these two aspects are necessary for enabling autonomous navigation and understanding the position of people (or intruders) in the environment. The object detection problem concerns establishing what types of objects are visible inside the camera field-of-view (FoV) and detecting their projection onto the 2D image plane as bounding boxes. The bounding box is defined by the coordinates of the rectangle' corner that contains all pixels belonging to a detected object as tightly as possible to its actual boundary shape. Therefore, object detection entails two complement sub-problems. The first is classifying image windows into a pre-established set of object categories (among which a *null* detection class is usually present). The second is the regression of two bounding box's corners x and y coordinates, determining their position on the image plane.

Following, we proceed to summarise the milestones in the object detection literature. Still, we refer to more extensive reviews of the state-of-the-art (SotA) [175, 330] for a more in-depth discussion. Moreover, Cazzato *et al.*. [33] provide an overview of object detection focusing on the UAV application.

As in localisation and mapping research history, initial object detection algorithms relied on hand-crafted features. For example, the Viola and Jones (VJ) face detection algorithm [293] obtained real-time performance by combining Haar features with the integral image technique to maximise the reuse of computations. Whereas the features enabled training Adaboost [85] to classify image windows into face's positive or negative detection, the VJ algorithm straightforwardly approaches the bounding box localisation with a sliding window of various scales and a detection cascade to reduce the computational overhead. Among other hand-crafted features, Dalal and Triggs [55] propose the Histogram of Gradients (HOG) feature descriptor to obtain invariance properties against translation, scale, illumination, and other transformations, thereby improving the robustness of the proposed detector. Before the stream of engineered approaches slowed their progress, the Deformable Parts Model (DPM) algorithm [80] and its numerous extensions [81, 82, 98] reached the peak of innovation. In more detail, DPM proposes dividing the problem by detecting parts of the complete object represented with a spring-based graph model. Then, it formulates a graph-matching problem that minimises the cost of part detection, still based on HOG features, and the cost of spring deformation.

The advancements of DL brought new perspectives to the field. Remarkably, after 2012 [148], the CNN models entirely replaced old approaches for feature extraction. At the same time, researchers improved the sliding windows technique by proposing various "region proposal" algorithms that speed up the detection process and achieve a higher recall rate. Among the most popular methods, the R-CNN method from Girshick *et al.*. [97] uses selective search [287, 291] to apply a CNN on a limited number of boxes (around 2000) and, from each selected box, extract features that a linear Support Vector Machines (SVM) could classify into an object category. Fast R-CNN [96] removes external classifiers and regressors and shares the CNN computations on common regions from selective search. To this end, it introduces a region of interest (RoI) pooling layer to direct the CNN feature map portion under the proposed box's scaled region to a fully connected layer for later object classification and bounding box regression. Faster R-CNN [227] further increases the speed of the region proposal step with a Region Proposal Network (RPN) optimisable with the CNN backbone and completes the transposition from classical object detection to a total DL approach. Finally, we mention Feature Pyramid Network (FPN) [172] since it has introduced a fundamental component replicated and improved by later methods. In detail, FPN provides an additional parallel structure to process feature maps at various scales. It consists of a feature pyramid of a top-down network laterally connected with the CNN backbone, which consents to obtain multi-resolution predictions at each pyramid level.

The methods above fall in the approach type called in the DL community "two-stage detection" because there is a clear distinction between a phase of region proposals filtering and a phase of actual object classification and bounding box position refinement. On the other side, "one-stage detectors" aim to simultaneously resolve the class prediction and bounding box regression with a unified architecture. The most renowned networks of this kind are You-Only-Look-Once (YOLO) and Single Shot Detector (SSD). YOLO [224] divides the image into a grid and for each cell predicts the probability of containing an object, its class, and the bounding box offsets with respect to an anchor box, *i.e.*, a box with a predefined aspect ratio, which shows the highest Intersection over Union (IoU) with the ground truth labels among a default set of anchors. Newer YOLO versions were introduced by the original authors [225, 226] or by other researchers inspired by their work [20, 179, 211] to extend the base design. SSD [177] is conceptually similar to YOLO but, as primary differences, substitutes the last fully connected layers with convolutions and performs multi-scale predictions using feature maps at different layers of the CNN. The main characteristic of one-stage is their impressive speed obtained by removing the burden of a RPN. In contrast, two-stage methods usually achieve higher detection accuracy. However, newer research publications narrowed this gap [187] and made one-stage networks more appealing, especially for robotics applications requiring lower latency times.

1.5 Objective, Scope, and Contributions

The objective of this thesis, as delineated in section 1.1, is to provide solutions for the perception of an autonomous aerial surveillance system. The definition of the problem allows some assumptions directed toward specific research directions. First, since the deployment environment consists of a confined area that we can survey beforehand, it is possible to exploit offline mapping techniques and use this knowledge for visual localisation in a second moment during the online operations. This offline procedure is the principal difference compared to SLAM approaches that require optimising a global map online without any clue of the scene scale or other a priori knowledge. Furthermore, we relax the requirements of continuous adaptation to dynamically changing scenarios as preached by the CL SLAM paradigm by maintaining constant environmental settings.

(a) DJI Mini 2 [83] (b) Ryze Tello [255] (c) Parrot Anafi 4K [141]

Figure 1.3: Example of commercial drones

The second assumption regards the sensory system. The intent is to rely entirely on a monocular RGB camera mounted on a UAV. This restriction is motivated by various factors dictated by commercial trends or research questions. Primarily, the recent regulations in Europe by the European Union Aviation Safety Agency (EASA) restrict the type of drone allowed to fly in urban areas based on weight categorisation. Hence, due to its weight, the presence of LIDAR is excluded from UAVs, which is the lowest category that allows flying in outdoor places with few precautions. Also, the high cost of LIDAR makes this type of sensor unsuitable for commercial drones designed for large customer audiences.

On the other hand, finding solutions that scale to more accessible hardware is of great interest when dealing with concrete problems, as recently demonstrated by Tesla's decision to transit from a Radio Detection and Ranging (RADAR) assisted autonomous navigation to pure vision driving autopilot [281]. For this reason, the type of drones depicted in Figure 1.3 represents the platform for which we are more inclined to develop a surveillance system. Considering the limited onboard computing capabilities of these UAVs, an external server is needed to run the algorithms for localisation and object detection. The new communication technologies that matured during the last years suggest that soon high bandwidth and low latency wi-fi channels will be available to offload the computations to edge servers entirely [110].

Furthermore, the approach sought to the robot perception does not exploit IMU data because of partial information disclosed by the commercial drone software development kit (SDK) or the possible absence of hardware synchronisation that would lower the complexity of multi-modal sensor fusion. Hence, this limitation engages the investigation for methods that rely purely on vision, similar to those reviewed in section 1.3. Finally, we distinguish proprioceptive sensors, like an IMU or a camera, from exteroceptive ones, like the GPS. While the firsts measure the robot's internal state from a self-centred perspective, the GPS gives the global position in the Earth-centred geographic coordinate system (GCS) using the signals of a satellite constellation. Since surveillance is susceptible to security threats, self-centred sensors guarantee higher protection against interference with the localisation algorithm. Instead, the GPS is sensible to jamming or spoofing attacks, [56] or the signal might become degraded or unavailable in certain areas [11]. Therefore, camera localisation provides a valid alternative in such cases.

Recalling the open perception questions related to our application context of aerial surveillance, in this thesis, we aim to find the answers to the research questions listed below.

- What deep learning approach is a valid alternative to more traditional approaches for solving ego-motion and depth estimation problems using only monocular images? How can we combine traditional motion estimation methods with deep learning to improve neural network predictions?

- How to design an efficient CNN model for predicting the UAV global pose, taking to our advantage the assumption of deploying the autonomous system in a controlled environment?

- How do dynamic objects, such as the presence of people in the camera FoV, influence the localisation predictions of deep learning models?

- How deep learning CNN is used to detect the people in colour images? Which information do we need to obtain the 3D position of the detected people in the environment?

- How to reconstruct the 3D scene for modelling an outdoor urban environment where the surveillance has to take place? How to create the data needed for training the DL models for localisation?

Therefore, this thesis develops multiple methodologies for solving these challenges, discussed in the chapters ahead and summarised next, bringing several contributions to the SotA. With these contributions, we seek to answer the previous research questions point-by-point.

- To address the first point, we present RAUM-VO, an algorithm to improve the pose estimates of unsupervised pose networks for monocular odometry. To this end, we introduce an additional loss to supervise the motion training. Moreover, we perform a rotation adjustment step combining the output of a traditional epipolar method with the predictions of the trained pose network. Finally, we compare our method with SotA approaches, either traditional or learned, on the widely adopted KITTI benchmark for odometry estimation.

- Then, by reviewing the literature of VBL, we identify, among the direct pose estimations paradigms, the end-to-end learning approach as the more efficient and well-suited to our objective. Specifically, we consider the assumption of a controlled and restricted environment as those for which the surveillance application is intended. Furthermore, we argue that a neural network trained for pose regression may well generalise to novel camera frames. Therefore, we introduce a neural network model composed of an efficient CNN for feature extraction and a MLP for the regression of the pose vector. Moreover, we compare multiple design options for the MLP regressor to discover improvements in the predictions. Finally, we compare our method with the close SotA approach on two datasets and evaluate the run-time performance on a GPU compute board for robotics applications.

- Subsequently, we investigate the impact of dynamic objects on the training of global pose regressors. To this end, we perform a statistical analysis based on the results of the ablation experiment. In detail, we use a CNN for object segmentation to pre-process the dataset by masking the objects. We use such data to compare the localisation performances with the standard training approach. Furthermore, we adopt techniques to visualise the contribution of each image pixel to the pose error in the form of saliency maps. With these tools, we can better understand the properties of the CNN models applied to the task of global localisation.

- Moving forward, we discuss the implementation of a DL approach for detecting objects appearing on colour images. Hence, we train an efficient SotA network leveraging distributed deployment techniques to minimise the optimisation times and fully exploit server cluster hardware. Moreover, we address the imbalance class issue caused by our particular use case of binary object classification. Whereas most datasets contain labels for multiple object types, we consider only the subset of people and modify the hyperparameters accordingly to maximise the detection accuracy. Lastly, we derive simple equations from projecting an object's 2D bounding box base central point to an estimated ground plane. We use this result to obtain the detected people's 3D position in a reconstructed environment.

- Finally, we study the incremental structure from motion (SfM) pipeline and a SotA implementation used to create the 3D model of the environment necessary for the people 3D localisation algorithm. Additionally, we align the reconstructed scene with approximate GPS coordinates of the registered camera frames to obtain a correct metric scale. Hence, we use SfM as an alternative to creating ground truth poses for learning the global and relative localisation tasks. Then, we reconstruct two outdoor urban environments to experiment with the proposed localisation and object detection algorithms. Remarkably, we introduce a novel scenario with images captured on the University of Luxembourg's Belval campus using only a lightweight commercial drone carrying a monocular camera. Finally, we show extensive results to prove the validity of the developed methodologies.

1.6 Thesis Structure

- In chapter 2, we describe the theoretical framework of DL, with details about the mathematics behind its optimisation and the CNN model principal characteristics and architectures used throughout the thesis.

- In chapter 3, we discuss the problem of visual odometry estimation using a monocular camera. Then, we propose a novel loss for guiding the training of an unsupervised neural network and an approach that integrates a traditional motion estimation with the network-predicted poses.

- In chapter 4, we address the problem of global metric localisation using an efficient CNN for feature extraction, and we compare different MLP designs to regress the pose vector. Finally, we analyse the impact of dynamic objects in the FoV of the camera on the network-predicted poses.

- In chapter 5, we tackle the intrusion detection problem in an outdoor urban area. First, we illustrate the computations involved in object detection that guide our implementation of an SotA network. Subsequently, we develop a method for localising a person in a 3D space relying on the object detection network and a fixed ground plane. Ultimately, while describing the theory behind 3D scene reconstruction, we list the steps to create a 3D model for two outdoor environments used for the experiments.

- In chapter 6, we conclude this thesis and discuss possible future extensions of this work.

2
Deep Learning Background

AI is the field that studies theories and techniques to develop software applications for programmable computers that can reproduce human behaviours and automate many challenging tasks. Machine learning (ML) is an AI subset of algorithms that aim to learn and improve from experience, which consists of a large number of data samples without an exact algorithm for the solution but with the sole guidance of the current learning performance metric. Mitchell [201] gives the following definition of a machine learning algorithm:

> "A computer program is said to learn from experience E for some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

The experience refers to the dataset available to learn a task and is assumed that the data-generating process produces independent and identically distributed (i.i.d.) samples. The dataset's data points are divided into training and test set. Differently from the training set, the test contains only unseen data points. However, following the i.i.d. assumption, both are described by the same underlying probability distribution.

The task can take various forms but is generally identified (also within the scope of this dissertation) with finding a relationship between input data and the desired output. This relationship can be approximated by a function $h_\theta : X \mapsto Y$ in the space $\mathcal{H}_\theta$ of the hypothesis parametrised by the model parameters θ. After training, h_θ can be used to predict new outputs for new inputs in the test set. When the result consists of a discrete set of categories, the task is called classification; otherwise, if the objective is to predict one or more numbers, *i.e.*, a vector, the task is called regression. An incomplete list of other known tasks comprehends machine translation, anomaly detection, synthesis and sampling, denoising, filling missing values, etc. In practice, a ML algorithm is a program that finds patterns in the data by optimising the model's parameter towards minimising prediction errors.

The performance is measured quantitatively by a "loss function" that expresses the error of an ML algorithm specific to the training task at hand. The error calculated on the training set is called the training error, whereas the test error refers to the prediction error measured on the test set. The first is how much the model fits the data points in the training set. The second indicates the performance of the model on novel unobserved inputs. The challenge of ML is to maintain the test error as low as the training, and this ability, called generalisation, is what distinguishes ML from optimisation algorithms [101].

Finally, we distinguish two main learning paradigms based on the type of information available in the dataset to guide the training process.

- Supervised learning supposes the availability of label or target data associated with the input data $\mathbf{x}$. Therefore, the ML algorithm is given a supervision signal $\mathbf{y}$, just like a teacher, that directs the learning towards the correct mapping between input and output. It is also straightforward to compute a measure of the model performance using some metric function of the distance between the ground truth labels $\mathbf{y}$ and the model prediction $\hat{\mathbf{y}}$.
- Unsupervised Learning aims to find underlying patterns in the data without explicit labelling. The ML algorithm should discover hidden properties of the input to form a compact abstract representation of the data that preserve the valuable information for solving the task. The loss function has to be designed to instruct the model to produce the required output without knowing the correct target. Therefore, it can be necessary to have more complex loss functions that replace the missing knowledge with a mathematical formulation of the task.

Among the ML algorithms, DL approaches AI problems with a powerful mathematical model called DNN.

2.1 Neural Networks Mathematics

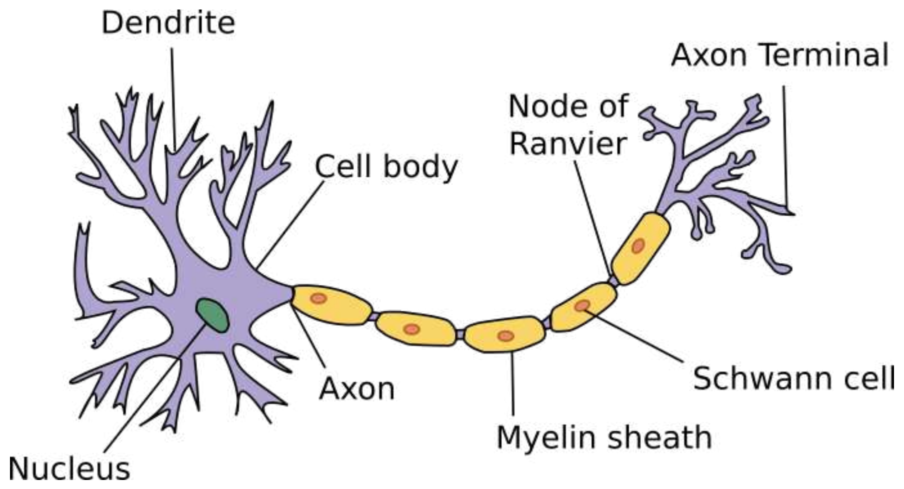


Figure 2.1: Abstraction of a neuron
An abstract representation of a biological neuron [68]. The Dendrite receives signals from other neurons. The Soma processes the information and computes a new signal. The axon transmits the signal through the Synapse that forms the point of connection with other neurons.

Neural networks have originated in the research on the human brain's functionality. McCulloch and Pitts [197] worked on the first mathematical models to simulate the biological activity of the neuron artificially (see Figure 2.1). Inspired by their work, Rosenblatt proposed "The Perceptron" [233] that showed the ability to learn from data by correcting the weights associated with the input connections. In Figure 2.2, a representation of an artificial neuron is presented. The neuron computes a weighted sum of all the inputs with an additional bias term that behaves as a threshold or prior and sets a minimum value when the input signal is zero. Then, a non-linear activation function g transforms the weighted sum and produces the final output of the neuron.

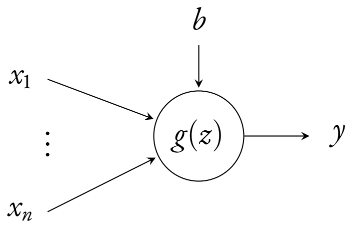


Figure 2.2: A model of an artificial neuron

Generated by editing code from [264].

The calculation performed by the neuron can be interpreted as a linear space transformation and a subsequent non-linear projection. The first operation is expressed by a multiplication between the weights' vector and the inputs' vector plus the bias term; the second by the application of an activation function as follows:

$$z = \mathbf{w}^\top \mathbf{x} + b\,, \tag{2.1}$$

$$y = g(z)\,, \tag{2.2}$$

where $\mathbf{w} \in \mathbb{R}^n$ is the vector of weights multiplied with to n inputs $\mathbf{x} \in \mathbb{R}^n$. Then, an ensemble of multiple neurons composes a layer of a network. Therefore, the operation of a single neuron becomes a matrix-vector multiplication:

$$\mathbf{z} = \mathbf{W}^\top \mathbf{x} + \mathbf{b}\,, \tag{2.3}$$

where $W \in \mathbb{R}^{m \times n}$ is the weights' matrix for m neurons connected to n inputs; $\mathbf{b} \in \mathbb{R}^m$ and $\mathbf{z} \in \mathbb{R}^m$ instead are the bias and respectively the output in vector form.

A neural network is composed of numerous layers other than input and output, called "hidden layers" because their output is not visible outside. We call a fully connected network of neurons, as represented in Figure 2.3, MLP. Each of the subsequent layers learns a non-linear decision boundary by projecting its input vector into a higher dimensional space whose size depends on the number of neurons. Intuitively, by getting deeper into the network layers, the MLP learns and refines an abstract representation of the data that is meaningful for the task. Recent techniques for visualising the weights of the hidden layers allow observing this insight experimentally by optimising each neuron input to maximise the activation function's *stimulus* [312]. Hence, the alternative and more general naming DNN recall the design of stacking layer after layer to process the data with networks that expand highly along the depth dimension and less along the width [113].

Finally, the "universal approximation theorem" [119] grasps the outstanding capability of DNN to address tasks that are difficult to solve with hard-coded programs. Notably, it states that a neural network can approximate any arbitrarily complex functions that are continuous on a compact domain [152]. This ability depends on the presence of a non-linear activation function in the hidden layers' neurons. Otherwise, we could reduce the network to a single linear projection. The choice of the activation function will influence the shape of the decision boundary created by the neurons and is one of the leading architectural design decisions.

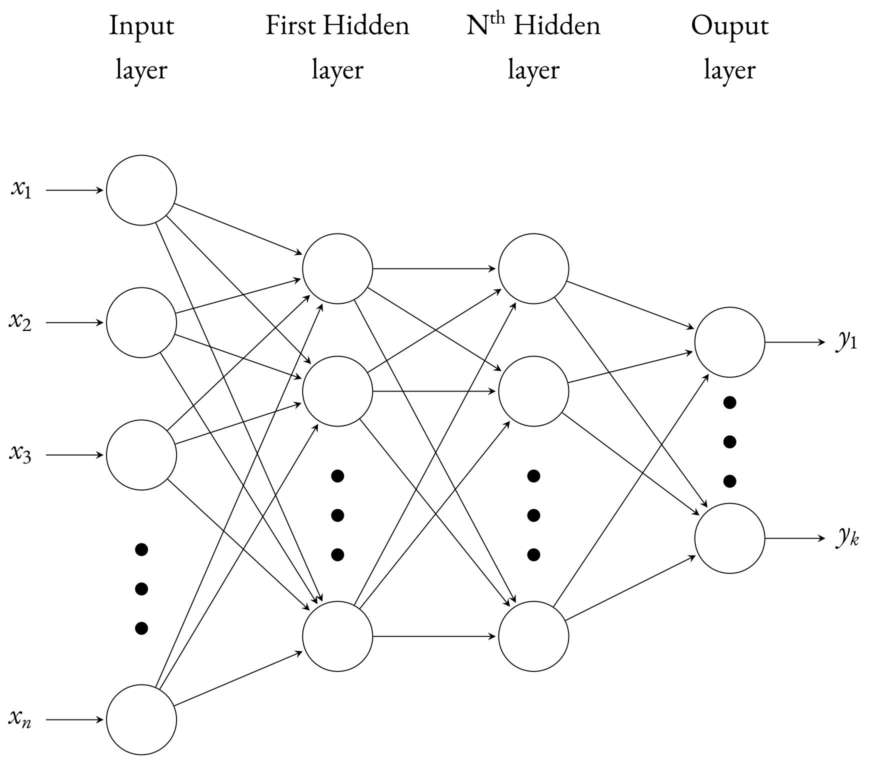


Figure 2.3: Example of a multi-layer perceptron (MLP)

Generated by editing code from [264].

2.1.1 Activation Functions

The initial versions of neural networks primarily used logistic sigmoid $\sigma(\cdot)$ or hyperbolic tangent $\tanh(\cdot)$ as an activation function. The properties of differentiability and continuity at all points were preferred for the learning process (see Figure 2.4). Also, they output a value bound between -1 and 1 or 0 and 1, which makes it intuitive to implement a network of logic gates. The two functions are defined as follows, where the $\tanh(\cdot)$ definition reuses the sigmoid expression to highlight their close relation:

$$\sigma(x) = \frac{1}{1 + exp(-x)} , \tag{2.4}$$

$$\tanh(x) = 2\sigma(x) - 1 . \tag{2.5}$$

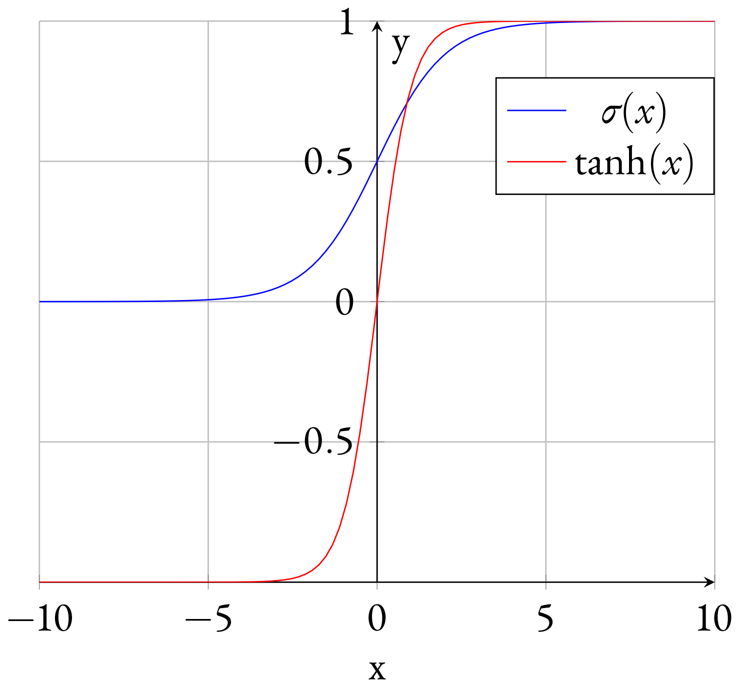


Figure 2.4: Logistic sigmoid and hyperbolic tangent functions

However, a downside of these functions is that they saturate quickly for high input values, *i.e.*, the gradients approach fast zero at the two extremes of the domain. Because the training process of a neural network is gradient-based (explained in the following Section), this characteristic could slow down the learning if not also prevent its progress completely. This issue motivates using different activation functions for the innermost hidden layers to help propagate the gradient up to the shallower.

The default design recommendation for recent implementations of DNNs is to use the Rectified Linear Unit (ReLU) activation function [3] (see Equation 2.6). ReLUs resemble closely linear functions and preserve most of its properties that make the gradient-based optimisation easier and generalise well for its simplicity, *e.g.*, the Occam's razor principle mentioned in subsection 2.1.3. It is said to be piecewise linear as it is composed of two pieces of a linear function connected at zero, where there is only discontinuity in the gradient. In practice, careful software implementations can solve the differentiability issue. So, for positive inputs, the ReLU has a positive and constant gradient, while for the rest of the domain is zero. This property captures the nature of biological neurons, which have a proportional response or are inactive, where the deactivation at certain inputs allows learning of sparse representations.

$$\mathrm{ReLU}(x) = max(x, 0) = \begin{cases} 0 & \text{if } x \leq 0\,, \\ x & \text{if } x > 0\,. \end{cases} \tag{2.6}$$

Following the ReLU, many variations have been proposed to address the absence of a gradient for negative inputs. Among the most popular, it is worth mentioning the ELU [51](see Equation 2.8), Leaky ReLU [191](see Equation 2.7), and Swish [222](see Equation 2.9), also called SiLU. Contrary to ReLU, these functions can also output negative values, and the Swish is non-monotonic, as visible in Figure 2.5. Also, they introduce a parameter α to control the shape of the function. However, they contribute to learning improvements only in restricted circumstances and tasks while adding computational cost.

$$\mathrm{LeakyReLU}(x) = \begin{cases} \alpha x & \text{if } x \leq 0\,, \\ x & \text{if } x > 0\,. \end{cases} \tag{2.7}$$

$$\mathrm{ELU}(x) = \begin{cases} \alpha(e^x - 1) & \text{if } x \leq 0\,, \\ x & \text{if } x > 0\,. \end{cases}, \tag{2.8}$$

$$\mathrm{Swish}(x) = \frac{x}{1 + e^{-\alpha x}}\,. \tag{2.9}$$

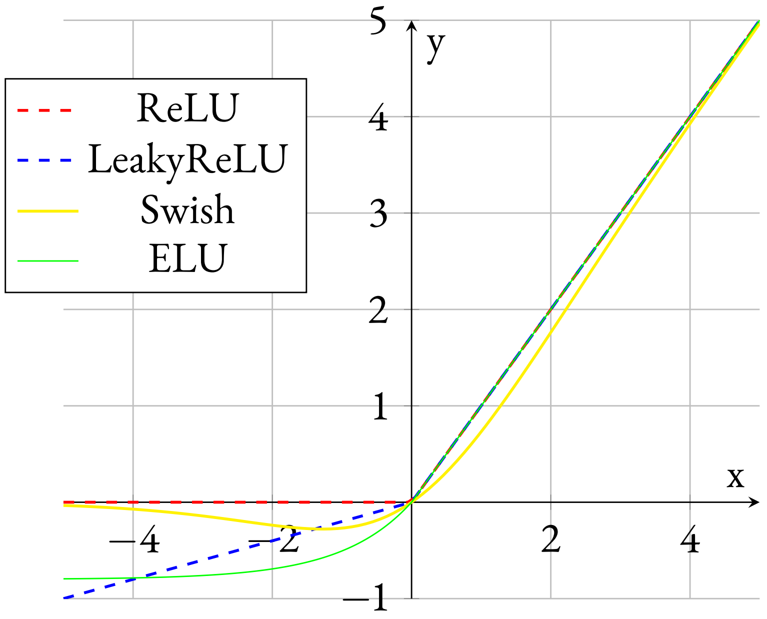


Figure 2.5: ReLU functions and their variations

2.1.2 Optimisation

The first component to design for learning a task is the loss function. The loss $J(\theta)$ is a scalar function parameterised by the set of network weights θ that produces a cost to be minimised. The cost is an expression of the network's prediction quality and is measured using the output $\hat{y} = h_\theta(\mathbf{x})$, where h_θ is the model approximation function, and the ground truth labels y in the case of a supervised task. For example, in the case of a regression task with known ground truth labels y, it is possible to compute an L_p loss between the two and reduce the average sum over N data samples:

$$J(\theta) = \frac{1}{N}\sum_{i}^{N} \|h_\theta(\mathbf{x}_i) - y_i\|_p = \frac{1}{N}\sum_{i}^{N} \|\hat{y}_i - y_i\|_p \,. \tag{2.10}$$

In general, the loss cost can be expressed as the expected value of a per-sample loss function $\mathcal{L}$, like the former L_p example, computed over the training dataset distribution p_{train}:

$$J(\theta) = \mathbb{E}_{(\mathbf{x},y)\sim p_{train}} \mathcal{L}\left(h_\theta(\mathbf{x}_i), y_i\right) = \frac{1}{N}\sum_{i}^{N} \mathcal{L}\left(h_\theta(\mathbf{x}_i), y_i\right) \,. \tag{2.11}$$

Hence, the learning process corresponds to finding the best DNN's parameters θ^* that minimise the loss cost value:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} J(\theta) \,. \tag{2.12}$$

The search for a (close to) optimal solution involves an optimisation algorithm tuning the parameters based on the loss improvement direction. In mathematical terms, this direction is found by computing the partial derivatives of the loss function J concerning each of the neurons' weights w_{ij}, *i.e.*, $\frac{\partial J}{\partial w_{ij}}$. Furthermore, the notion of gradient $\nabla_\theta J(\theta)$ generalises that of partial derivatives with θ representing the set of all network parameters or weights w_{ij}.

The calculation of the gradients is the most expensive operation and is performed in three steps:

1. A "forward propagation" of the inputs produces the output values for all neurons until the end of the network.
2. The loss cost is computed using the final predictions and the ground truth labels if they are available.
3. The "back-propagation" (shortly "backprop") step computes the gradients by building a computational graph and leveraging the chain rule of derivatives.

The graph tracks the flow of information through the network and creates nodes for each operation applied to the input to produce the final output. Also, the graph is needed to compute the chain of derivation efficiently by following a specific computation order and minimising the waste of operations. The chain rule of calculus is a basic approach to breaking down complex function derivative formulations into other functions for which it is simpler to compute the derivatives. In general, the rule states that the derivative of a composed function $z = f(g(x))$ with $y = g(x)$ is:

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}\,. \tag{2.13}$$

For a complete treatment of backpropagation algorithms, the reader is referred to Chapter 6 of the book *Deep Learning* [101].

So, once the gradient is obtained, it is possible to optimise the weights by descending along the curvatures of the loss cost surface [165]. Since the gradient points to cost increases, we perform a step in the opposite direction to minimise it. Hence, the following weights' update rule represents the optimisation method known as steepest descent or batch gradient descent:

$$\theta = \theta - \eta \nabla_\theta J(\theta) = \theta - \eta \sum_{i=1}^{N} \nabla_\theta \mathcal{L}_i\,, \tag{2.14}$$

where N is the size of the training set and $\nabla_\theta \mathcal{L}_i$ is the gradient of the per-sample loss with respect to the network parameters θ computed on the i-th training data point. The quantity η is called the learning rate and is the primary hyperparameter to tune for setting convergence speed. The update is performed until a stopping condition is reached, *e.g.*, the loss is less than a certain ε, or when several training "epochs" have been completed. The term "batch" suggests using the entire training dataset in every update step. Therefore, the optimisation process is deterministic and becomes quickly intractable as the memory requirements grow with larger datasets.

Due to the non-linear nature of a DNN model, the optimisation problem is certainly non-convex. Thus, it is impossible to guarantee that the optimisation algorithm converges to a globally optimal solution but more probably lands on a local minimum or a saddle point. However, it is possible to obtain improved solutions using wiser optimisation strategies. The first fundamental measure is randomly initialising the weights with small positive random values because of the flat or low gradient signal in the negative part of the activation functions' domain. Instead, the biases may also be zero and sometimes are not included among the parameters at all. Secondly, it is possible to introduce some degree of stochasticity in the learning process by considering "mini-batches", *e.g.*, using small batches of data and not the entire training dataset, which helps escape local minima and find a larger convergence basin. Stochastic Gradient Descent (SGD) is an extension of the basic gradient descent in Equation 2.14 that performs the update step using only a randomly sampled subset of the training set, a mini-batch. It is called stochastic because it estimates an approximation of the real gradient as the average gradient of the mini-batch:

$$\theta = \theta - \eta \nabla_\theta J(\theta) = \theta - \eta \sum_{i=1}^{B} \nabla_\theta \mathcal{L}_i \,, \tag{2.15}$$

where B is the batch size.

Notably, Keskar *et al.*. [138] discuss the role of small batch size in escaping "sharp minimisers" [117], which are points of convergence for which the cost increases rapidly in their neighbourhood, and that show weak generalisation properties. However, increasing the batch size is fundamental to speeding up learning on large datasets. In such cases, adjusting the learning rate η is required to maintain the generalisation capabilities. With this regard, Goyal *et al.*. [103] present a warmup strategy and suggest the following rule to modify the learning rate in relation to the batch size :

> **Linear Scaling Rule**: when the mini-batch size is multiplied by k, multiply the learning rate by k.

From a probabilistic perspective, the trade-off between the batch size and learning rate influences the noise scale that controls the fluctuations in the gradient descent dynamics [259]. This noise scale g is expressed as:

$$g = \eta \left(\frac{N}{B} - 1 \right) \,, \tag{2.16}$$

where N is the number of samples in the training set, and B is the batch size. In general, the noise level has to be gradually reduced by applying a learning rate decay schedule that ensures a stable point of convergence is reached and that respects the following sufficient conditions [231]:

$$\sum_{i=1}^{\infty} \eta_i = \infty \text{ and} \tag{2.17}$$

$$\sum_{i=1}^{\infty} \eta_i^2 < \infty \,. \tag{2.18}$$

Various scheduling functions are commonly used in practice, from linear to exponential or even step decay. Alternatively, following the previous considerations on the noise scale, Smith *et al.*. [258] propose to increase the batch size for exploiting training on multiple distributed GPUs.

2.1.3 Regularisation

Due to the increase of layers in deeper models neural networks, the maximum complexity of the learned approximation function tends to augment proportionally to the number of parameters. The ability of a model to represent a wide hypothesis space is defined as capacity [101]. When a model cannot grasp the mapping between the input training data and its label, the training error does not decrease further after reaching a minimum plateau line. In this case, the model is said to underfit the training data, and this issue can be overcome by altering its capacity to apprehend more about the hidden properties of the data. In contrast, when the model expresses a large capacity in relation to the quantity of training data to analyse, we can incur the adverse challenge of overfitting. If the gap between the training and the test error is large, a model is said to overfit the training set as its generalisation error is high. This scenario occurs when the model selects an approximation function in the hypothesis space that perfectly fits the data points of the training set, but it fails to grasp unseen samples. Occam's razor principle conveys the idea that among the set of viable hypotheses, the model should choose the least complex that best explains the data points. Regularisation techniques address the issue of generalisation error by modifying the learning algorithm to control the complexity of the model.

Parameter Norm Penalty , or weight decay, works by adding to the loss function $J(\theta)$ a constraint on the norm of the network's parameters, thus limiting the capacity of learned models effectively, as in the following Equation:

$$\tilde{J}_{\theta} = J_{\theta} + \alpha \|\theta\|_{p} , \tag{2.19}$$

where p represents the degree of the norm defined on the parameter space, and α is a hyperparameter controlling the penalty magnitude. The L_2 norm is frequently preferred, referred to as ridge regression or Tikhonov regularisation in this context, which encourages evenly small weights by affecting more outliers. Another option is the Lasso regression, *i.e.*, L_1 penalty, which invokes sparsity by pushing the norm of some parameters to zero. It has been extensively applied as a form of feature selection with simpler linear models or SVM.

Dropout [260] is a form of regularisation that deactivates, or drops out, a subset of the neurons (see Figure 2.6) selected randomly from a Bernoulli distribution with probability p. Also, it makes the learning noisy, making it harder for the network to overfit the training data. Furthermore, Dropout limits the co-adaptation problem that prevents the neurons from learning independently from each other and instead encourages learning sparse representations. Finally, since only a portion of the neuron is active during each update, Dropout reduces the actual capacity of the hypothesis space. However, for this reason, the more nodes are shut down, the larger the network has to be to maintain the same test accuracy, and longer training times are required.

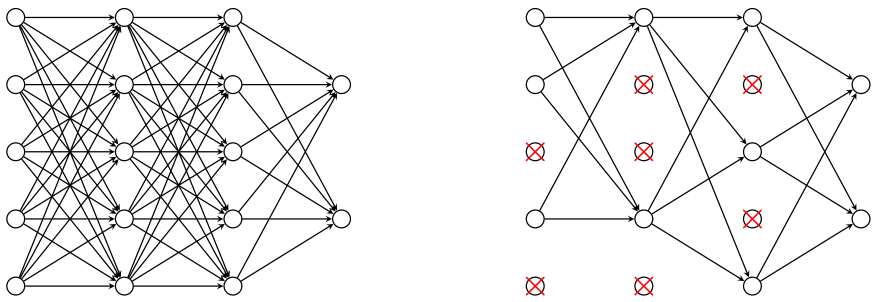


Figure 2.6: Dropout effect visualisation

On the left, a neural network with two hidden layers. On the right, the same network after applying dropout. The red crossed nodes have been randomly selected to be shut off. Image inspired by the original paper [260]. Generated by using the code from [230].

Batch Normalization [124] was not designed to be a regularisation method *per se* but to make the training easier and faster by reducing the "Internal Covariate Shift" effect. This issue is due to the assumption that weights not involved in calculating the gradient for the update step remain constant while simultaneously optimising all the network parameters. Hence, the activation's output distribution moves uncontrolled without the hidden layers being aware of this variation, which causes deep model training to slow down. Batch normalization addresses the problem through an adaptive reparametrisation of the hidden layers' inputs. First, it tracks the mean $\mu_{\mathcal{B}}$ and the standard deviations $\sigma^2_{\mathcal{B}}$ of the mini-batch $\mathcal{B}$ linear transformation $x_i^{(k)}$ computed by the k-th layer's neurons:

$$\mu_{\mathcal{B}} = \frac{1}{B}\sum_{i=1}^{B} x_i^{(k)}\,, \tag{2.20}$$

$$\sigma^2_{\mathcal{B}} = \frac{1}{B}\sum_{i=1}^{B} (x_i^{(k)} - \mu_{\mathcal{B}})^2\,. \tag{2.21}$$

Following, the input to each layer's activation function is normalized, or "whithened", by the following operation:

$$\hat{x}_i^{(k)} = \frac{x_i^{(k)} - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \varepsilon}}, \tag{2.22}$$

where B is the size of the mini-batch, and ε is a small constant for numerical stability. Then, the network learns two parameters, *i.e.*, the new variance $\gamma^{(k)}$ and mean $\beta^{(k)}$ of the batch distribution, which shift and scale the input of the previous layer in a regulated manner and are required for preserving the expressive capability:

$$y_i^{(k)} = \gamma^{(k)} * \hat{x}_i^{(k)} + \beta^{(k)} . \tag{2.23}$$

The noise introduced by Batch Normalisation provides an additional source of regularisation that reduces the amount of Dropout or even makes it unnecessary. This operation is applied by default in most modern DNN architectures for its effectiveness while being elegant and easy to implement.

2.2 Convolutional Neural Network

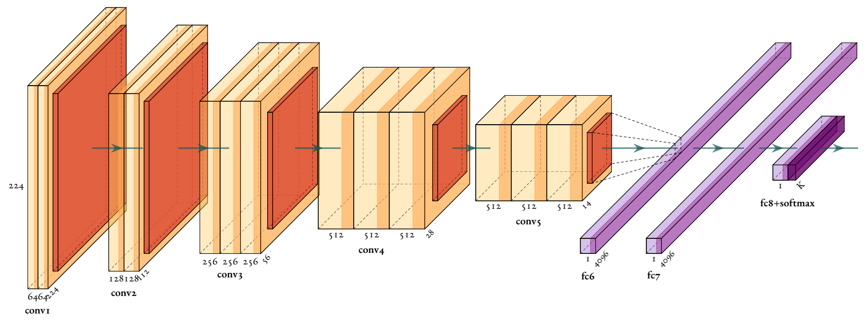


Figure 2.7: Example of a CNN architecture

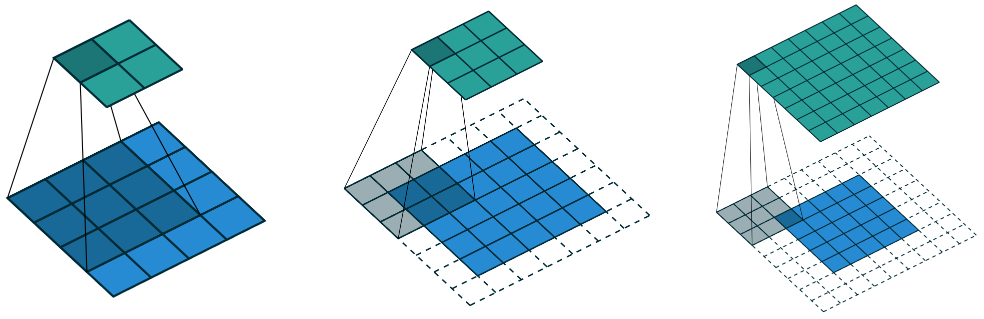
The figure illustrates a VGG16 [256] CNN model. At its end, 3 fully connected layers, part of and MLP, and a Softmax function layer are connected to output a probability score. This plot was generated by using the code from [125].

Reasoning about the complexity of the MLP network design, it is noticeable that the fully connected layers' complexity scales quadratically with the input size. This computational cost hinders the application of neural networks to computer vision due to the high dimensionality of images. The DNN research expanded with the introduction of the CNN technique that settled the premises for advancing in all those tasks that required visual data analysis. The LeNet architecture [154] represents the first breakthrough advancement in deep CNNs applied to handwritten character recognition. With a lower memory footprint and a computationally effective implementation on GPU hardware, the CNN approach obtained a similar error rate to SVM. However, even if, in the subsequent years, CNN started winning challenges in various tasks, the establishment of CNN as the best approach for computer vision problems arrived with the publication of AlexNet [148]. The authors proposed a network with 60 million parameters, shared between five convolutional layers and three fully connected layers. This surpassed the previous SotA approaches and drastically reduced the image classification error on the challenging ILSVRC-2010 dataset [14]. Furthermore, AlexNet demonstrated that deeper architectures could achieve impressive results, paving the road for the research toward large networks. The VGG model (represented in Figure 2.7) enhances this design principle of creating bigger models towards the depth dimension and enlarging the width. This architecture uses mainly 3×3 kernels and doubles the channels after halving the feature maps spatial dimension with max pooling. With this elegant and simple design, VGG has been one of the most popular architectures able to achieve high performance in the image classification task.

2.2.1 CNN Basic Operations

Herein, we briefly describe the two operations that make most of the computation in a standard CNN architecture, convolution and pooling. Then, Dumoulin and Visin [70] provide a complete treatment of the CNN's arithmetics, to which we refer the reader to know more about the topic.

Convolution Operation

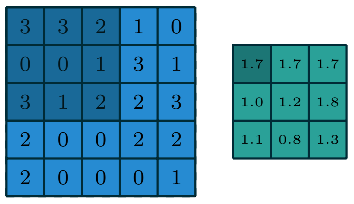


(a) Valid convolution (b) Same convolution (c) Full convolution

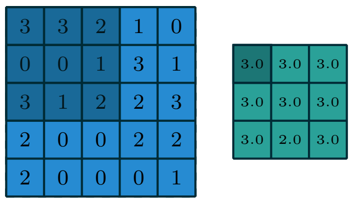
Figure 2.8: Convolution operation types compared
Figure openly available from the Github repo related to [70].

The convolution operation is a sparse linear transformation, which uses only a portion of the input in a local neighbourhood, and preserves the notion of adjacency [70]. Figure 2.8 provides an example of three types of convolutions. The blue grid is the input matrix, also called the feature map, whereas the shaded region represents the convolution kernel, *i.e.*, a matrix of learned weights. These are shared for all the locations of the input feature maps over which the kernel is passed. In fact, as in the standard convolution on 1D inputs for signal processing, the kernel slides over the input, and, for each overlapping position, a corresponding output value is computed. The output feature map, coloured in green, is created by an element-wise multiplication of the kernel with the underlying input portion and summing up the result, represented with a darker green colour. Based on the input and kernel size, the output size may also vary by adding some padding values around the input feature map. Besides the kernel size, the convolutions are characterised by the stride, which consists of kernel sliding move step length. The number of single kernels learned at each layer of a CNN corresponds to the channel dimension. Each kernel then learns a separate function, or filter, which can recognise a specific pattern in the image of feature maps and hallucinate abstract high-dimensional feature representations.

Pooling Operation



(a) Average Pooling



(b) Max Pooling

Figure 2.9: A numerical example of the avg. and max pooling operations
Figure openly available from the Github repo related to [70].

Besides the convolutional blocks, CNN is frequently made up by adding the pooling operation to shrink the spatial dimensions of the input features size, *i.e.*, the width and height. This operation allows computing convolutions on smaller inputs, which, in turn, permits expanding the depth. Pooling applies a function, typically the average or the max, to condense the information contained in sub-regions of the input features map. Hence, pooling works similarly to convolution kernels with a sliding window (without learned parameters usually) that feeds the underlying values to the pooling function [70]. In Figure 2.9, a numerical example compares the result of average pooling (on the left) with max pooling (on the right). Finally, pooling is also characterised by its kernel size and stride parameters.

2.2.2 CNN Architectures

Herein, we describe some of the most relevant CNN architectures. These have already been mentioned in our survey regarding object detection methods for UAV [33] as an efficient network architecture that is suitable for feature extractions in real-time robotics applications. For this reason, we recall some principal characteristics that motivate their application in the algorithms introduced.

MobileNet [121] is built upon the concept of depthwise separable convolutions, first introduced by Sifre & Mallat [254] and popularised by the Xception network [47]. This computation module consists of one first layer of kernels that operate on each input channel independently, the depth-wise convolution, followed by the point-wise that combines the intermediate feature maps through a 1×1 convolution. MobileNet controls the number of output channels at the end of each block with a hyperparameter named width multiplier.

MobileNetV2 [238] adds linear bottleneck layers at the end of the separable convolutions to create what the authors call inverted residual block since the skip connection with the identity feature maps is performed when the network shrinks the number of channels. The authors believe ReLU non-linearity can preserve the information manifold contained in the channels when it lies in a low-dimensional subspace of the input.

Neural Architecture Search (NAS) is a novel, prolific field whose purpose is to use search algorithms, usually either Reinforcement Learning or Evolutionary optimisation methods, to find a combination of modules that obtains an optimal trade-off between latency and accuracy. For instance, the optimisation metric could be specified as a Pareto multi-objective function as demonstrated in MnasNet [270]. FBNet [304], ChamNet [54], and MobilenetV3 [120] are other examples of successive contributions to mobile network architecture search. The reader can refer to [74, 303] for a thorough analysis of this technique.

EfficientNet [271] improves the ability to manually tune the model complexity letting the users choose the desired trade-off between efficiency and accuracy. Pointedly, the authors propose a compound scaling hyperparameter that adjusts the network depth, *i.e.*, the number of layers or blocks, the width, *i.e.*, the number of channels, and the input image size, which influences the inner feature maps' resolution, all at once in an optimal way. This scaling method follows the observation that the network's dimensions do not independently influence the latency-accuracy trade-off. Finally, this novel scaling concept is applied to a baseline network found by taking advantage of the multi-objective optimisation and search space specified in MnasNet.

3
Unsupervised Ego-Motion and Depth Estimation

In this chapter*, we discuss the problem of visual odometry estimation using a monocular camera. Unsupervised learning for monocular camera motion and 3D scene understanding has gained popularity over traditional methods, which rely on epipolar geometry or non-linear optimization. The unsupervised training protocol is similar to the direct SLAM methods. Both approaches synthesize a time-adjacent frame by projecting pixel intensities using the current depth and pose estimations and minimizing a photometric loss function. However, the learned strategy differs from the traditional one because the network incrementally incorporates the knowledge of the 3D structure and the possible range of motions into its weights, giving better hypotheses during later training iterations. Moreover, deep learning can overcome many of the typical issues of traditional monocular visual odometry. For example, the support of a large amount of example data during training can help solve degenerate motions (*e.g.*, pure rotational motion), scale ambiguity and scale drift, initialization and model selection, low or homogeneously textured areas, and perceptual aliasing [268]. In addition, concerning supervised learning, we can fully leverage video stream data without needing depth or motion labels. However, being aware of the solid theory behind the traditional methods [107] and their more general applicability, we leverage geometrical image alignment to improve the pose estimation. Herein, we note that rotational motion can limit the accuracy of the unsupervised pose networks more than the translational component. Therefore, RAUM-VO is introduced to address this issue. RAUM-VO, shown in Figure 3.1, is an approach based on a model-free epipolar constraint for F2F estimation (shortly named F2F) to adjust the rotation during training and online inference. To this end, we match 2D keypoints between consecutive frames using pre-trained deep networks, while training a network for depth and pose estimation using an unsupervised training protocol. Then, the predicted rotation is adjusted with the motion estimated by F2F using the 2D matches and initializing the solver with the pose network prediction. Ultimately, RAUM-VO shows a considerable accuracy improvement compared to other unsupervised pose networks on the KITTI dataset, while reducing the complexity of other hybrid or traditional approaches and achieving comparable state-of-the-art results.

*The work presented in this chapter has been published in the *Sensors* journal [48] and reproduced herein with few modifications.

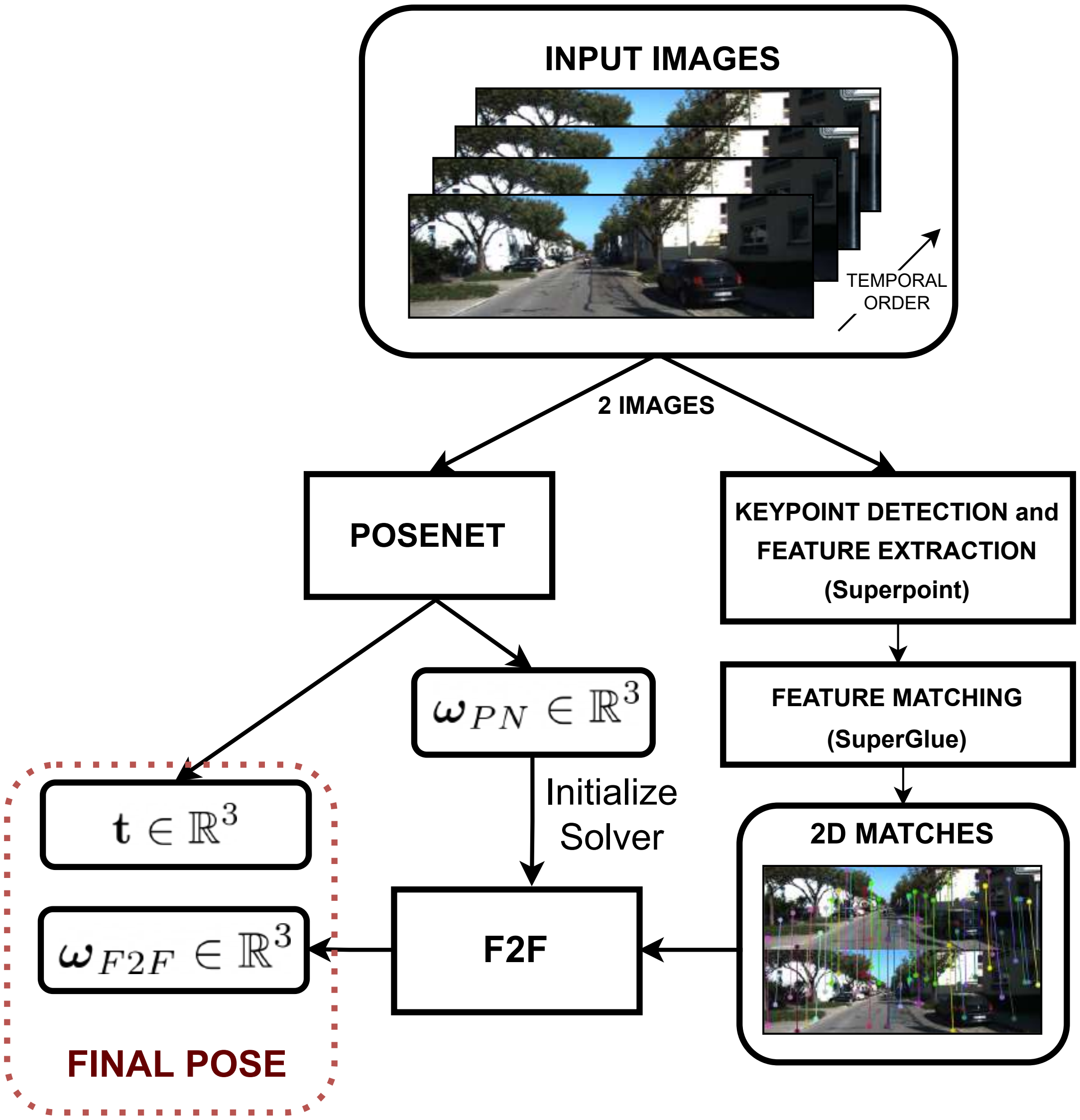

INPUT IMAGES
TEMPORAL ORDER
2 IMAGES
POSENET
KEYPOINT DETECTION and FEATURE EXTRACTION (Superpoint)
FEATURE MATCHING (SuperGlue)
2D MATCHES
$\omega_{PN} \in \mathbb{R}^3$
Initialize Solver
F2F
$\mathbf{t} \in \mathbb{R}^3$
$\omega_{F2F} \in \mathbb{R}^3$
FINAL POSE

Figure 3.1: RAUM-VO block diagram.
The figure shows the flow of information inside RAUM-VO from the input image sequence to the final estimated pose between each pair of consecutive image frames.

3.1 Literature Review

Unsupervised Learning of Monocular VO

The pioneering work of Garg *et al.*. [88] represents a fundamental advancement because they approached the problem of depth prediction from a single frame in an unsupervised manner for the first time. Their procedure synthesises a camera's depths in a rectified stereo pair by warping the other using the calibrated baseline and focal lengths. Godard *et al.*. [99] use the stereo pair to enforce a consistency term between left and right synthesized disparities while adopting the Structural Similarity Metric (SSIM) metric [299] as a more informative visual similarity function than the L_1 loss. SfM-Learner [324] relies entirely on monocular video sequences and proposes the use of a bilinear differentiable sampler introduced with the Spatial Transformer Network (STN) [127] to generate the synthesized views.

Because the absolute metric scale is not directly observable from a single camera (without any prior knowledge about object dimensions), stereo image pairs are also helpful to recover a correct metric scale during training while maintaining the fundamental nature of a monocular method [100, 166, 313]. Mahjourian *et al.*. [192] impose the scale consistency between adjacent frames as a requirement for the depth estimates by aligning the 3D point clouds using iterative closest point (ICP) and approximating the gradients of the predicted 6 DoF transform. Instead, Bian *et al.*. [16], arguing that the previous approach ignores second-order effects, show that it is possible to train a globally consistent scale with a simple constraint over consecutive depth maps, allowing one to reduce drift over long video sequences. In [190], a SfM model is created before training and used to infer a global scale, using the image space distance between projected coordinates and OF displacements. More recently, several approaches [167, 314, 320] have leveraged learned OF dense pixel correspondences to recover up-to-scale two-view motion based on epipolar geometry. Therefore, they resolve the scale factor by aligning a sparse set of points with the estimated depths.

One of the main assumptions of the original unsupervised training formulation is that the world is static. Hence, many works investigate informing the learning process about moving objects through OF [30, 43, 129, 157, 164, 189, 223, 292, 298, 311, 316, 329]. TheOF, which represents dense maps of the pixel coordinates displacement, can be separated into two components. The first, the rigid flow, is caused by the camera's motion. The second, the residual flow, is caused by dynamic objects that move freely in relation to the camera frame. Therefore, these methods train specific networks to explain the pixel shifts inconsistent with the two-view rigid motion. However, these methods focus principally on the depth and OF maps quality and give few details about the impact of detecting moving objects on the predicted two-view motion. Notably, they use a single metric to benchmark the relative pose that is barely informative about the global performance and cannot distinguish the improvements clearly.

A recent trend is to translate traditional and successful approaches such as SVO [84], LSD-SLAM [77], ORB-SLAM [203], and DSO [76] into their learned variants, or to take them as inspiration for creating hybrid approaches, where the neural networks usually serve as an initialization point for filtering or pose graph optimization [17, 44, 169, 182, 283, 297, 307, 308]. However, RAUM-VO focuses on improving the predicted two-view motion of the pose network without introducing excessive computation overhead as required by a PGO backend.

Instead of training expensive OF, RAUM-VO leverages a pre-trained Superpoint [66] network for keypoint detection and feature description and Superglue [240] for finding valid correspondences. Unlike OF, the learned features do not depend on the training dataset and generalize to a broader set of scenarios. In addition, using Superglue, we avoid heuristics for selecting good correspondences among the dense OF maps, which we claim could be a more robust strategy. However, we do not use any information about moving objects to discard keypoints lying inside these dynamic areas. Finally, differently from other hybrid approaches [314, 320], we do not entirely discard the pose network output, but we look for a solution that improves its predictions efficiently and sensibly.

Pure Rotation Problem in Monocular VO

Pure rotational motion is the main problem in monocular VO and SLAM because depth cannot be inferred by triangulation and the epipolar constraint becomes numerically intractable when the translation vector norm approaches zero [145, 268]. Many SLAM approaches, as for example ORB-SLAM [203] in its initialization phase, handle this issue using the geometric robust information criterion (GRIC) score introduced by Torr *et al.*. [284] for model selection. In practice, they decide when is the case to apply a homography transformation that can correctly model the 3D rotation.

A complementary technique found in the literature of keyframe-based indirect SLAM is that of relying on panorama maps and on deferred feature registration by tracking points at the infinite plane or rays until a large baseline is found for the triangulation. Gauglitz *et al.*. [90] were the first to adopt panoramas but their use is limited to describe a wider range of motions. However, as noted in [217], they do not use the information contained in panoramas for augmenting the global 3D map. Therefore, Pircheim *et al.*. [217] create a single hybrid map in which points with finite and infinite depth are combined and used for optimizing the reprojection error. When enough correspondences are found between finite and infinite points, panorama keyframes are relocalized with respect to the 3D map and new infinite features can be triangulated. In DT-SLAM [25], 2D and 3D features contributes together to the pose estimation and BA formulations. Frames showing pure rotation constrain the translation to be zero and non-triangulated features are mapped along with the others so that their depth estimation can be deferred until enough parallax is observed. Zhou *et al.*. [326] expose the pose estimation flaws of LSD-SLAM [77] when rotation-only motion occurs. Instead, they adopt the Bayesian framework to model the depth observations first presented in [295], which distinguishes between good and unknown measurements with a probability π that balances the distribution for the two kinds.

A more recent approach to the pure rotation motion problem is that of decoupling it from the translation component and estimating it in isolation. Once relative rotations, possibly noisy, are obtained for multiple camera positions, the problem of finding n optimal absolute orientations consistent with the relative estimates is formalized as *rotation averaging* [109] and several methods have been proposed to solve it [38, 39, 62, 79, 108, 194]. Carlone *et al.*. [29] evidence that 3D pose graph SLAM would reduce to a linear least-square problem in the case the rotations were known. That being said, they survey five techniques to initialize the pose graph with optimized rotations and show the improved solutions and convergence times. In [24], a variant of Trimmed ICP [45] is used to align back-projected feature rays, and the method of Chatterjee *et al.*. [38] is adopted to refine the estimation by solving the rotation averaging problem. Chng *et al.*. [46] propose an incremental formulation of rotation averaging. To estimate the relative rotations, they use the novel method of Kneip and Lynen [144] who propose an alternative epipolar constraint based on normal vectors and find the rotation by solving an iterative eigenvalue minimization problem independently from the translation. Recently, Lee and Civera [158] extend this method to multiple views proposing an approach that preserves the properties of BA and rotation averaging, thereby deserving the name of rotation-only BA.

Contrary to the traditional VO methods, unsupervised neural networks can improve the ego-motion estimation even in the presence of pure rotational motion by predicting the depth maps along with the relative poses and transferring prior learned knowledge of the scene to the most difficult frame sequences. Additionally, from the literature on geometrical frame-to-frame motion estimation methods, we select the model-free epipolar constraint of Kneip and Lynen [144] to find the best rotation between two subsequent camera views. Remarkably, the whole set of input matches can be modelled at once with this approach without resorting to multiple motion models and RANSAC iterative fitting loops.

3.2 Unsupervised Learning of Depth and Ego-Motion

This section outlines the proposed algorithm, RAUM-VO, for estimating the motion from a sequence of monocular camera images using a combination of deep neural networks and traditional epipolar geometry. This work follows Zhou *et al.*. [324], who established an unsupervised training protocol based on view synthesis and photometric loss, which we describe in subsection 3.2.1. In addition, to facilitate the learning process, we describe additional techniques implemented in our training in subsection 3.2.2 and subsection 3.2.3. As shown in Figure 3.2, the training outcome is a depth network that has learned to associate a disparity map to a single input image frame and a pose network that predicts the 6 DoF rigid transformation between two consecutive frames. Additionally, we use the Superpoint [66] network to extract 2D keypoints descriptors. Consequently, using a pre-trained Superglue Graph Neural Network (GNN) [240], RAUM-VO matches the corresponding features between pairs of successive frames. These matches are the input for the two-view motion estimation method [144] (see section 3.3), whose rotation corrects the network's output.

3.2.1 View Synthesis and Photometric Loss

The principle for obtaining a supervision signal is similar to direct visual odometry [297]. Given two images at time t and $t+1$, I_t and I_{t+1}, respectively, the depth network produces disparity (inverse depth) maps d_t and d_{t+1}, respectively, and the pose network produces a 6 DoF transformation.

$\mathbf{T}_{t\to t+1} = [\,\mathbf{R}\,|\,\mathbf{t}\,]$.

Then, we obtain the depth maps D_t and D_{t+1} by inverting the disparities and normalizing them between a predefined minimum and maximum range limit. Finally, let $\mathbf{K}$ denote the intrinsic camera matrix, and $\mathbf{p}_t = [u, v]$ a 2D pixel coordinate on I_t image plane, in 2D homogeneous coordinates. The projection of $\mathbf{p}_t$ into the reference frame of I_{t+1}, $\mathbf{p}_{t\to t+1}$, is given by the following equation:

$$\mathbf{p}_{t\to t+1} = \pi(\mathbf{K}\mathbf{T}_{t\to t+1}\mathbf{K}^{-1}\mathcal{H}(\mathbf{p}_t, D_t[\mathbf{p}_t]))\,, \tag{3.1}$$

where $D_t[\mathbf{p}_t]$ denotes the depth value at the point $\mathbf{p}_t$, and $\mathcal{H}$ is the operation to lift the 2D pixel coordinates to 3D homogeneous coordinates:

$$\mathcal{H}\colon ([u, v], z) \mapsto [u * z, v * z, z, 1] = [x, y, z, 1]\,, \tag{3.2}$$

while π is the projection to the image plane:

$$\pi\colon ([x, y, z, 1]) \mapsto [x/z, y/z] = [u, v]\,. \tag{3.3}$$

Using the (sub-)differentiable bilinear sampling operation, which we note with $\mathcal{S}$, introduced with the STN [127], we obtain a synthesized version of I_{t+1}, $I_{t\to t+1}$, by interpolating its intensity values at the locations indicated by a grid of points $\mathbf{p}_{t\to t+1}$.

$$I_{t\to t+1} = \mathcal{S}(I_{t+1}, \mathbf{p}_{t\to t+1})\,. \tag{3.4}$$

Next, we optimize the estimated disparities and poses by minimizing the perceptual distance between the image I_{t+1} and its synthesized version $I_{t\to t+1}$. Following the initial suggestion of [317] and the example of previous similar works [99, 166], this distance is best assessed by a combination of L1 and SSIM [299], which is differentiable concerning both depth and pose networks parameters. Particularly, the SSIM function aims to quantify the visual similarity of I_{t+1} and it's synthetic reconstruction $I_{t\to t+1}$ by comparing the luminance, contrast, and structure measurements on windows of size $n \times n$.

Therefore, the *photometric loss* L_p, equates to:

$$L_p = \alpha_{\text{SSIM}} \frac{1 - \text{SSIM}(I_{t+1}, I_{t\to t+1})}{2} + \alpha_{l_1} \|I_{t+1} - I_{t\to t+1}\|_1 . \tag{3.5}$$

In our experiments, we set $\alpha_{\text{SSIM}} = 0.85$ and $\alpha_{l_1} = 0.15$.

Notably, this warping mechanism succeeds with the assumption that the scene is static, there are no occlusions, and the lighting conditions are constant, without reflections. Notwithstanding that the training process may be robust to minor violations of these assumptions, solutions for reducing dynamic objects [189] and non-Lambertian surfaces' [307] impact on the optimization convergence have been provided in the recent literature. Instead, we rely on simpler mechanisms to alleviate the dynamic world conditions. During training, we extend the view synthesis procedure to the previous frame I_{t-1} as well. Hence, we consider the minimum between $L_p(I_{t-1}, I_t)$ and $L_p(I_{t-1}, I_t)$ on a per-pixel basis as the final photometric loss. This strategy mitigates the effects of disoccluded pixels [100].

To conclude, we would like to add a few observations. First, while the output would be random at the beginning, it is expected to converge to a meaningful value through the joint optimization process of the two networks. Next, the scale of the 6 DoF transformation, foreseeably, reflects the depth scale, as they are jointly optimized. However, even if not aligned with the metric scale of the scene, it is plausibly globally consistent. Remarkably, this is an advantage over geometrical methods since, for the latter, we would need to take further precautions to avoid scale drifts [77, 261]. In subsection 3.2.3, we will introduce an additional loss term to reinforce a global consistency constraint during training.

3.2.2 Depth Smoothness Loss

The photometric loss is not informative with homogeneous or low-textured areas of an image, and the depth estimation problem becomes ill-posed. The pixels in these regions can be associated with disparity values and still obtain a similar visual appearance for a fixed rigid transformation [100]. However, we can introduce a prior on the estimated depth maps that encourage smooth changes of the disparities inside these regions while discouraging the formation of holes. Thus, by considering the first (or second [297])-order gradients of the image as weighting terms, we allow sharp discontinuities to appear only in correspondence of edges [99].

Therefore, the following equation constitutes the *depth smoothness loss* L_s:

$$L_s = |\partial_x d_t| \, e^{-|\partial_x I_t|} + \left|\partial_y d_t\right| e^{-\left|\partial_y I_t\right|}, \tag{3.6}$$

where ∂_x and ∂_y are the first derivatives of the colour image and disparity map taken along x and y directions.

3.2.3 Depth Consistency Loss

An issue of monocular VO, famously, is the non-observability of the metric scale of the surrounding environment and, consequently, the motion between two views. This limitation leads to the well-known issue of scale drift, which has been successfully addressed in traditional BA-SLAM by performing the pose graph optimization over 3D similarity transforms [77, 261]. From the perspective of learned monocular VO, Tateno *et al.*. [277] explores the path of predicting depth maps using CNN, confident of their capability to reproduce the metric scale passed through the ground-truth depths supervision. On the other hand, without depth supervision, an alternative approach to learning a metrically scale-aware network is from information regarding the translation vectors norm, as in [106], where the authors impose a velocity loss. Even though we cannot obtain the real scale during training, ensuring depth consistency is fundamental for reducing the drift and easing the task of aligning the estimated trajectory with an external metric map. Therefore, in this work, lacking the knowledge of real-world scale and ground-truth depths, we adopt the loss for imposing depth consistency between two frames introduced by Bian *et al.*. [16]. The following equation defines the *depth consistency loss* L_{dc}:

$$L_{dc} = \frac{|D_{a\to b} - D_b|}{D_{a\to b} + D_b}\,, \tag{3.7}$$

where $D_{a\to b}$ represent the synthesized version of the depth estimated for image I_a to the camera reference of image I_b through the estimated pose $\mathbf{T}_{a\to b}$ and the bilinear sampler.

3.3 Geometrical Loss and Rotation Adjustment

Here, we describe the pivotal component of our proposed method. In particular, we incorporate the rotation optimization formulated by Kneip and Lynen [144]. They offer an alternative epipolar constraint that enables one to solve the relative pose problem without many of the issues encountered in essential-matrix-based methods. Namely, these are:

- the indirect parametrization of the motion that has to be decomposed from the essential matrix, as in [107]:

$$\mathbf{E} = [\mathbf{t}]_x\,\mathbf{R}\,; \tag{3.8}$$

- multiple solutions from the decomposition that have to be disambiguated through a cheirality check and hence by triangulation;

- degenerate solutions that may result from either point lying on a single planar surface, distribution of the points in a small image area, and pure translational or rotational motion. In these cases, one approach is to select a different motion model, *e.g*. the homography matrix, after identifying the degeneracy with a proper strategy.

Therefore, given a set of image points $(\mathbf{p}_i, \mathbf{p}'_i)$ matched between two views, we translated them into pairs of unit-bearing vectors $(\mathbf{f}_i, \mathbf{f}'_i)$ through normalization. These vectors ideally start from the camera centre and point in the direction of the corresponding 3D points, and each pair defines an epipolar plane. Then, the authors observe that all the normal vectors of the epipolar planes need to be coplanar [145]. The normal vectors form together a 3-by-*n* matrix $\mathbf{N} = [\mathbf{n}_1 \ \dots \ \mathbf{n}_n]$, and are defined as follows:

$$\mathbf{n_i} = \mathbf{f_i} \times \mathbf{R}\mathbf{f}'_\mathbf{i} \, . \tag{3.9}$$

Due to the coplanarity constraint, the covariance matrix $\mathbf{N}\mathbf{N}^\top = \mathbf{M}$ has to be at most of rank 2. Notably, the problem is equivalent to a rank minimization parametrized by $\mathbf{R}$, and is solved by finding the matrix $\mathbf{M}$ with the smallest minimum eigenvalue:

$$\mathbf{R} = \arg\min_{\mathbf{R}} \lambda_{\mathbf{M},\min} \, . \tag{3.10}$$

Furthermore, the authors observe that the eigenvector associated with $\lambda_{\mathbf{M},\min}$ corresponds to the translation direction vector. Therefore, this method, which we name **F2F**, is able to retrieve the full frame-to-frame motion.

The problem is solved with a LM procedure. To avoid the possible presence of local minima typical of non-linear optimization, we use the rotation estimated by the pose network as a starting point. In Section subsection 3.6.1, we show the benefits of this initialization. In addition, we choose to perform a single optimization with all the matches instead of multiple RANSAC iterations. For restricting the number of matches outliers, we set the threshold of the Superglue match confidence score to 0.9. At the moment, we found that this approach works best for the data at hand after empirical evaluation of multiple RANSAC settings and inlier criteria.

Lastly, we include the rotation $\mathbf{R}_{\mathbf{f2f}}$ as supervision for the rotation output of the pose network, $\mathbf{R}_{\mathbf{PN}}$, in the *residual rotation loss* L_r. To this aim, we map the rotation matrices into their axis-angle counterparts through the logarithm function:

$$\log\colon \mathrm{SO}(3) \rightarrow \mathfrak{so}(3); \ \mathbf{R} \mapsto \log(\mathbf{R}) \, , \tag{3.11}$$

where $\mathfrak{so}(3)$ is the Lie algebra associated to the Lie group of 3D rotations $SO(3)$ [87]. Based on the isomorphism between $\mathfrak{so}(3)$ and $\mathbb{R}^3$ with the cross product, we treat the logarithm of a rotation matrix as a vector $\omega \in \mathbb{R}^3$ decomposed into a unit-norm direction vector $\mathbf{u} \in \mathbb{R}^3$, representing the rotation axis, and its L_2 norm $\theta \in \mathbb{R}$, where $\theta \in [0, \pi]$ represents the angle of rotation:

$$\log(\mathbf{R}) = \omega = \theta\mathbf{u}\,. \tag{3.12}$$

Therefore, we can compute the L_1 norm, denoted by $||\cdot||_1$, of the distance between the rotation vector predicted by the network, ω_{PN}, and the one estimated by F2F, ω_{f2f}. Thus, we obtain the following *residual rotation loss* L_r:

$$L_r = ||\omega_{F2F} - \omega_{PN}||_1\,. \tag{3.13}$$

The implementation of F2F used in this work is the one provided by the OpenGV library [143].

3.4 Networks Architectures

The depth network has an encoder-decoder architecture [232] with skip connection similar to DispNet [196] used by SfM-Learner [324]. Specifically, the encoder is a ResNet18 [114], and the decoder has five layers of 3×3 convolutions followed by an Exponential Linear Unit (ELU) activation function [51], an up-sampling, and a concatenation with the "connected" encoder feature. In accordance with [16], we avoid multi-scale training for efficiency purposes. Therefore, we apply the sigmoid function to the last output to obtain a disparity map.

The pose network consists of one ResNet18 [114] encoder that inputs a pair of images concatenated along the channel dimension. The feature extracted by the last layer is then the input to a small CNN decoder composed by:

1. one linear layer that reduces the feature to a 256-dimensional vector followed by the ReLU [3] non-linear activation function;
2. two convolutional layers with 256 kernels of size 3×3 followed by ReLU;
3. one linear layer that outputs the 6 DoF pose vector as the vector $\mathbf{x} \in \mathbb{R}^6$, which contains the concatenation of the translation $\mathbf{t} \in \mathbb{R}^3$ and the axis-angle rotation $\omega_{PN} \in \mathbb{R}^3$.

The network architectures are based on the Monodepth2 implementation [100] and use PyTorch [210]. Both network encoders are initialized with pre-trained weights on the ImageNet dataset [63].

3.5 Experiments

This Section provides details regarding our experimental procedure and the settings for accurately reproducing our results. In addition, we provide the results of VO obtained on KITTI and compare them with state-of-the-art methods.

3.5.1 Training Procedure

Because we have experienced a degradation in performance when including the l_{dc} term early in training, we split it into two phases. Particularly, when the depth network has not yet found a convergence direction for a plausible geometrical structure, the l_{dc} term, especially if it has a magnitude outweighing the photometric loss norm, could cause the depth maps to collapse towards a local minimum during the initial training phase. An alternative solution may be to adaptively adjust the weighing of l_{dc} based on the value of l_p. Therefore, we add the depth consistency loss after the convergence of the photometric loss. In addition, we add the contribution of the loss l_r in the second training phase to let the pose network reach an initial convergence plateau first. In Figure 3.2, we show how all the components we described interact during the training of RAUM-VO.

Consequently, we obtain two models:

- **Simple-Mono-VO** is obtained after the first training phase by selecting the checkpoint with the best t_{err} on the training set;
- **RAUM-VO** is obtained after the second phase by selecting the checkpoint with the best t_{err} on the training set and correcting the rotations with the output of F2F.

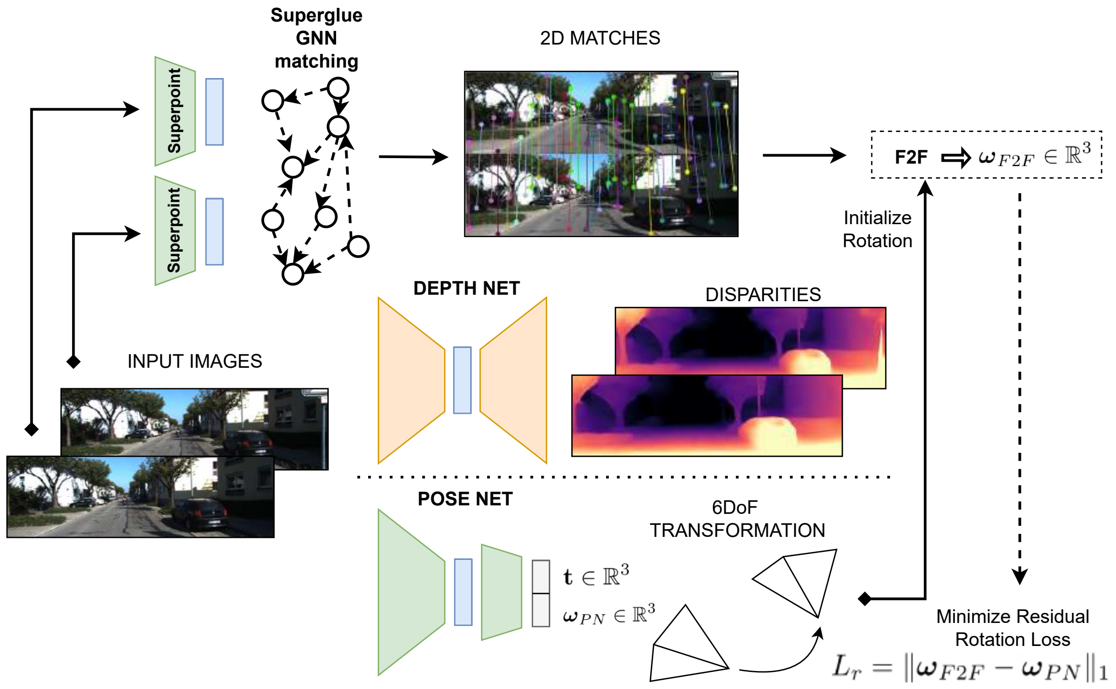

Superglue GNN matching
2D MATCHES
Superpoint
Superpoint
F2F ⇒ $\omega_{F2F} \in \mathbb{R}^3$
Initialize Rotation
DEPTH NET
DISPARITIES
INPUT IMAGES
POSE NET
6DoF TRANSFORMATION
$\mathbf{t} \in \mathbb{R}^3$
$\omega_{PN} \in \mathbb{R}^3$
Minimize Residual Rotation Loss
$L_r = \|\boldsymbol{\omega}_{F2F} - \boldsymbol{\omega}_{PN}\|_1$

Figure 3.2: Diagram of RAUM-VO training.

A sequence of images and 2D matches between pairs is the input for the training. The depth network takes only a single image to output a disparity map. The pose network outputs the 3D rigid transformation, as rotation and translation, between the two input images temporally ordered concatenated along the channel dimension. The matches are the input to the frame-to-frame rotation algorithm, whose output guides the training and adjusts the pose network estimation at test time.

3.5.2 Training Settings

The images are resized to 640×192 before entering the network. During training, we sample with repetition 2000 images for each epoch. We use standard colour image augmentation by slightly changing saturation, brightness, contrast, and hue, as in [100], and horizontal flipping. For the optimization, we use Adam [139] with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and a learning rate $lr = 10^{-4}$. We halve the learning rate when the loss does not decrease for 10 epochs. We keep the training until convergence of the loss or for at most 1800 epochs. The *depth smoothness loss*, *depth consistency loss*, and *residual rotation loss* weighing factors are 10^{-3}, 5×10^{-1}, and 1, respectively.

3.5.3 Dataset and Metrics

We evaluate our visual odometry network on the KITTI odometry dataset [91]. In fact, despite not being recorded from a UAV, this dataset currently represents the most popular benchmark for VO and SLAM in challenging urban settings. To this aim, we use sequences from 0 to 8 for training and sequences 9 and 10 for testing. Furthermore, we use the tool provided by the author of DF-VO [314] to ensure we apply the same criteria for evaluation. Notably, we evaluate with the "7 DoF alignment" setting that computes the similarity transform that best aligns the predicted trajectory with the ground truth using the Umeyama algorithm [288]. With this, we obtain a set of estimated poses $P = \{\mathbf{p}_i \in \mathrm{SE}(3) | i \in \mathcal{F}\}$ where $\mathcal{F} = \{1, 2, \ldots n\}$ is the set of frames in a sequence. Similarly, $Q = \mathbf{p}_{i_{i \in \mathcal{F}}}$ indicates the set of ground-truth poses for each frame.

The principal metric used to evaluate odometry is the Absolute Trajectory Error (ATE) that measures how consistent the estimated trajectory is with the real path. First, we compute the residual pose between pairs of frames using the inverse pose composition operator [151]:

$$e_i = q_i \ominus p_i \, , \; \forall i \in \mathcal{F} \, . \tag{3.14}$$

The the ATE is the Root Mean Square Error (RMSE) of the translational component error terms [219]:

$$\mathrm{ATE} = \left(\frac{1}{n} \sum_{i=1}^{n} \|\mathrm{trans}(e_i)\|^2 \right)^{\frac{1}{2}} . \tag{3.15}$$

Secondly, the Relative Pose Error (RPE), which is composed of the translational error in meters and the rotational in degrees, measures the drift by computing the average error of estimated motion between each frame interval. Precisely, we obtain the RPE of the rotational part as follows:

$$\mathrm{RPE}(^\circ) = \frac{1}{n} \sum_{i=1}^{n} \angle[(q_{i+1} \ominus q_i) \ominus (p_{i+1} \ominus p_i)] , \tag{3.16}$$

where $\angle[\cdot]$ gets the rotation angle from the pose matrix. Instead, the RPE for the translation is obtained using the euclidean norm of the translation vectors:

$$\mathrm{RPE}(m) = \frac{1}{n} \sum_{i=1}^{n} \|\mathrm{trans}((q_{i+1} \ominus q_i) \ominus (p_{i+1} \ominus p_i))\|_2 . \tag{3.17}$$

.

Finally, while the previous metrics are general for the majority of visual odometry benchmarks, in KITTI the authors introduce a variant of the translation and rotation errors by computing RPE for all possible subsequences of length (100,...,800) meters and averaging them at the end [92].

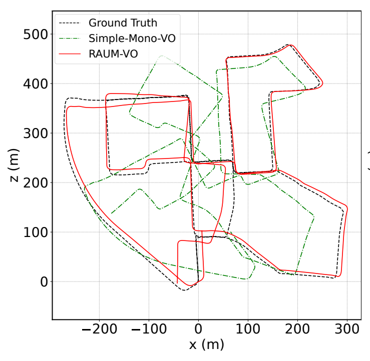
3.5.4 Results

In Table 3.1, we compare our results with two geometrical approaches, ORB-SLAM [203] and VISO2 [93]; two unsupervised networks methods, SfM-Learner [324] and the variant from Bian *et al.*. with depth consistency, SC-SfMLearner [16]; and with a hybrid approach DF-VO [314]. We note that the reported results for SC-SfMLearner are slightly different from the one in the paper and may refer to training with additional data. For our evaluation, we select those works that use only monocular image sequence during training and evaluation phases, as RAUM-VO does, because stereo image pairs give an unfair advantage to the depth reconstruction and, consequently, to the pose estimation, as documented in the literature [100]. Another condition for the evaluation regards the architectures of the depth and pose networks. Therefore, we selected methods in the learned categories that use comparable, not equal, deep networks. Unfortunately, this is one element of discrepancy among the works in the literature on the unsupervised pose and depth estimation, and that has to be considered when making comparisons. While RAUM-VO does not surpass DF-VO performances in many sequences, its accuracy is comparable while being more efficient. Because DF-VO is one of the most promising hybrid approaches using monocular images for the VO, in subsection 3.6.2, we examine the differences and advantages of our method in more detail. Regarding traditional methods, the average error of RAUM-VO is generally lower, except for the r_{err} metric computed on ORB-SLAM only. However, unlike ORB-SLAM, we do not apply local BA. Regarding the unsupervised pose networks category, the proposed RAUM-VO reduces the error effectively with the proposed rotation adjustment step.

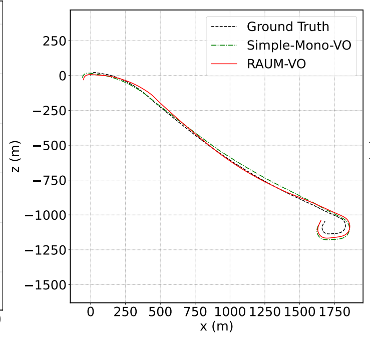
Table 3.1: Odometry quantitative evaluation on KITTI odometry seq. 00–10.

Best results are highlighted in **bold**, second best with an underline. Data is partially retrieved from [314].

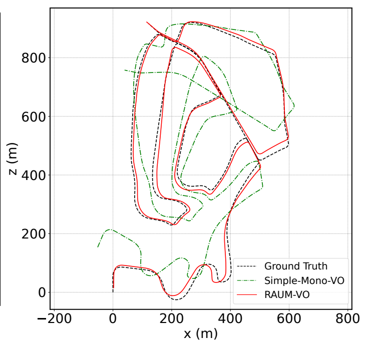
Category	Method	Metric	00	01	02	03	04	05	06	07	08	09	10	Train Avg. Err.	Tot. Avg. Err.
Geometric	ORB-SLAM2 [203] (w/o LC)	t_{err}	11.43	107.57	10.34	**0.97**	**1.30**	9.04	14.56	9.77	11.46	9.30	2.57	19.604	17.119
		r_{err}	**0.58**	0.89	**0.26**	**0.19**	**0.27**	0.26	**0.26**	0.36	**0.28**	0.26	**0.32**	**0.372**	**0.357**
		ATE	40.65	502.20	47.82	**0.94**	**1.30**	29.95	40.82	16.04	43.09	38.77	5.42	80.312	69.727
		RPE (m)	0.169	2.970	0.172	0.031	0.078	0.140	0.237	0.105	0.192	0.128	0.045	0.455	0.388
		RPE (°)	0.079	0.098	0.072	0.055	0.079	0.058	0.055	0.047	0.061	0.061	0.065	0.067	0.066
	VISO2 [93]	t_{err}	10.53	61.36	18.71	30.21	34.05	13.16	17.69	10.80	13.85	18.06	26.10	23.373	23.138
		r_{err}	2.73	7.68	1.19	2.21	1.78	3.65	1.93	4.67	2.52	1.25	3.26	3.151	2.988
		ATE	79.24	494.60	70.13	52.36	38.33	66.75	40.72	18.32	61.49	52.62	57.25	102.438	93.801
		RPE (m)	0.221	1.413	0.318	0.226	0.496	0.213	0.343	0.191	0.234	0.284	0.442	0.406	0.398
		RPE (°)	0.141	0.432	0.108	0.157	0.103	0.131	0.118	0.176	0.128	0.125	0.154	0.166	0.161
Unsupervised	SfM-Learner [324]	t_{err}	21.32	22.41	24.10	12.56	4.32	12.99	15.55	12.61	10.66	11.32	15.25	15.169	14.826
		r_{err}	6.19	2.79	4.18	4.52	3.28	4.66	5.58	6.31	3.75	4.07	4.06	4.584	4.490
		ATE	104.87	109.61	185.43	8.42	3.10	60.89	52.19	20.12	30.97	26.93	24.09	63.956	56.965
		RPE (m)	0.282	0.660	0.365	0.077	0.125	0.158	0.151	0.081	0.122	0.103	0.118	0.225	0.204
		RPE (°)	0.227	0.133	0.172	0.158	0.108	0.153	0.119	0.181	0.152	0.159	0.171	0.156	0.158
	SC-SfMLearner [16]	t_{err}	11.01	27.09	6.74	9.22	4.22	6.70	5.36	8.29	8.11	7.64	10.74	9.638	9.556
		r_{err}	3.39	1.31	1.96	4.93	2.01	2.38	1.65	4.53	2.61	2.19	4.58	2.752	2.867
		ATE	93.04	85.90	70.37	10.21	2.97	40.56	12.56	21.01	56.15	15.02	20.19	43.641	38.907
		RPE (m)	0.139	0.888	0.092	0.059	0.073	0.070	0.069	0.075	0.085	0.095	0.105	0.172	0.159
		RPE (°)	0.129	0.075	0.087	0.068	0.055	0.069	0.066	0.074	0.074	0.102	0.107	0.077	0.082
	Simple-Mono-VO (Ours)	t_{err}	9.365	8.920	6.830	3.697	2.570	4.964	3.138	3.568	7.125	13.625	11.131	5.575	6.812
		r_{err}	2.840	0.562	1.582	2.478	0.566	2.083	0.959	1.866	2.608	3.146	4.784	1.727	2.134
		ATE	94.949	30.004	83.155	4.112	2.377	30.227	8.726	8.872	59.887	66.591	18.792	35.812	37.063
		RPE (m)	0.090	0.304	0.087	0.037	0.055	0.041	0.051	0.044	0.074	0.166	0.077	0.087	0.093
		RPE (°)	0.072	**0.042**	0.057	0.048	0.036	0.049	0.040	0.048	0.052	0.067	0.083	0.049	0.054
Hybrid	DF-VO [314] (Mono)	t_{err}	**2.33**	39.46	3.24	2.21	1.43	**1.09**	**1.15**	**0.63**	**2.18**	**2.40**	**1.82**	5.969	5.267
		r_{err}	0.63	**0.50**	0.49	0.38	0.30	**0.25**	0.39	**0.29**	0.32	**0.24**	0.38	0.394	0.379
		ATE	**14.45**	117.40	19.69	1.00	1.39	**3.61**	**3.20**	**0.98**	**7.63**	**8.36**	**3.13**	18.817	16.440
		RPE (m)	**0.039**	1.554	0.057	**0.029**	**0.046**	**0.024**	**0.030**	**0.021**	**0.041**	**0.051**	**0.043**	0.205	0.176
		RPE (°)	**0.056**	0.049	**0.045**	**0.038**	**0.029**	**0.035**	**0.029**	**0.030**	**0.037**	**0.036**	**0.043**	**0.039**	**0.039**
	RAUM-VO (Ours)	t_{err}	2.548	**8.354**	**2.578**	3.217	2.860	3.045	3.033	2.390	3.632	2.927	5.843	**3.517**	**3.675**
		r_{err}	0.775	0.868	0.582	1.334	0.645	1.153	0.837	1.037	1.074	0.318	0.683	0.923	0.846
		ATE	16.272	**23.748**	**16.139**	2.602	2.283	17.470	9.234	2.164	16.303	8.664	12.297	**11.802**	**11.561**
		RPE (m)	0.040	**0.257**	**0.050**	0.030	0.052	0.038	0.046	0.028	0.053	0.068	0.078	**0.066**	**0.067**
		RPE (°)	0.059	0.062	0.048	0.048	0.035	0.044	0.042	0.058	0.045	0.042	0.051	0.049	0.049



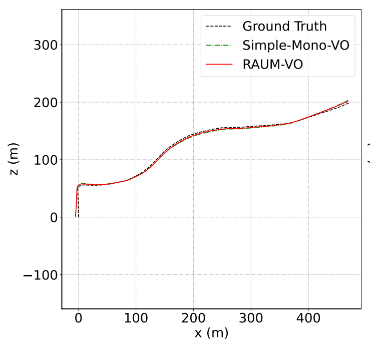
(a) Seq. 00



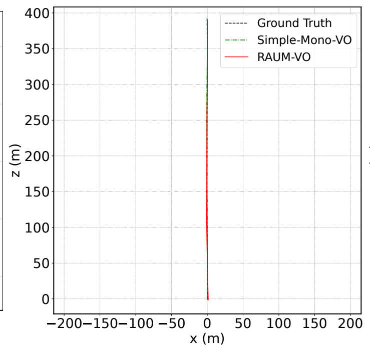
(b) Seq. 01



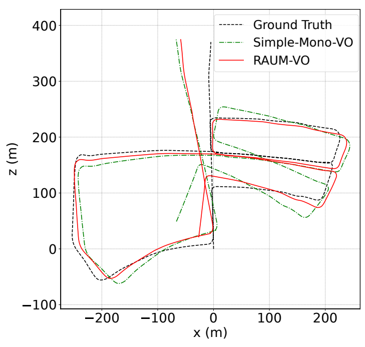
(c) Seq. 02



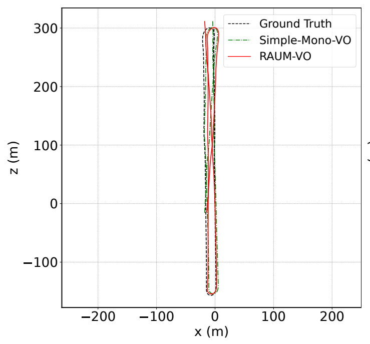
(d) Seq. 03



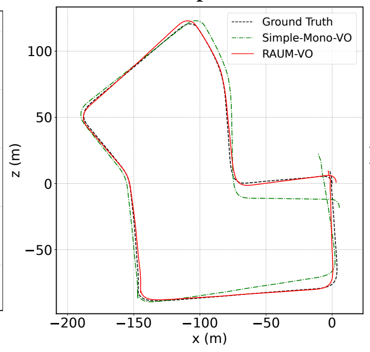
(e) Seq. 04



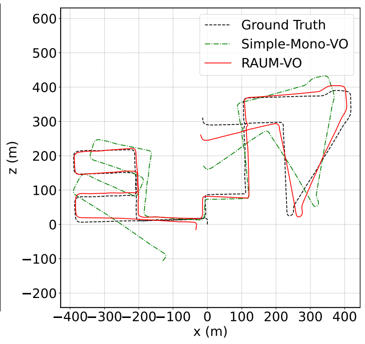
(f) Seq. 05



(g) Seq. 06



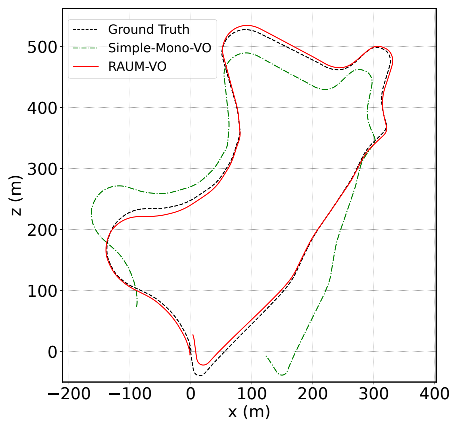
(h) Seq. 07



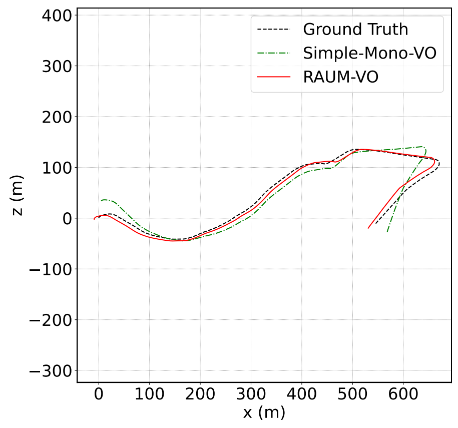
(i) Seq. 08

Figure 3.3: KITTI train trajectories

Estimated trajectories for the KITTI odometry sequences from 00 to 08. Poses are given in camera frame. Thus, positive *x* means right direction and positive *z* means forward. Best viewed in color.



(a) Seq. 09



(b) Seq. 10

Figure 3.4: KITTI test trajectories

Estimated trajectories for the KITTI odometry sequences 09 and 10. Poses are given in camera frame. Thus, positive x means right direction and positive z means forward. Best viewed in color.

In Figure 3.3, we show the plots of the trajectories for the training sequences predicted by our two models and the ground-truth poses. By comparing these with the testing sequences displayed in Figure 3.4, we can appreciate the generalization capability of the neural network to unseen sequences, even if KITTI contains images from similar scenarios. In the link (`https://youtu.be/4woTiJRCrUI`), a video that shows the depth map predictions for all the KITTI sequences is provided.

3.6 Discussion

Herein, we discuss and analyze the characteristics of RAUM-VO. First, in subsection 3.6.1, we consider the rotational and translational components of the pose error separately to argue that the rotations offer a larger space to decrease the absolute trajectory error (ATE) shown in Table 3.1. In turn, this motivates the adoption of a specific measure to adjust the predicted rotations. Hence, we demonstrate how the pose network plays a valuable role in initializing the F2F solver. Lastly, in subsection 3.6.2, we speculate on the factor that contributes the most to the accuracy of DF-VO compared to our approach.

3.6.1 General Considerations

In Table 3.2, we show that by modifying the simple-mono-VO predictions using the ground truth of either the translation or the rotation, there is a larger margin for improvement enclosed in the current rotation estimates than in the translational component of the error. We presume that this behavior is because we optimize translations directly on their vector space, contrary to the rotations. The manifold of rotations, *special orthogonal* group $SO(3)$, only locally resembles a Euclidean topology [156] and needs intermediate representations to enable the optimization with gradient descent methods. As such, the axis-angles are a many-to-one mapping with $SO(3)$, and alternative representations may be easier to approximate with a neural network [325]. In addition, the linear distance metric between translation vectors is easier to approximate than the non-linear counterparts for the $SO(3)$ group [123]. Nevertheless, the rotation provided by the pose network is a better initialization point for the F2F than the identity or constant motion assumption. The results of the different types of initialization are visible in Table 3.3. By this, the pose network's predicted rotations are always the best option for initializing the F2F solver and are paired only by constant motion assumption in some cases.

Then, we suggest that the pose network can regress the motion even in difficult motion situations, assuming that the depth network has learned a valid geometric structure. The pose and depth outcomes are strongly entangled due to their joint training, even if produced by separate networks. However, more precisely, we note that the performance of one component may be restricted by the other. While this may seem a trivial conclusion, it is necessary to clarify the limitations of this approach and bring us to the last reflection. We evaluate the odometry poses obtained by PnP combined with the depth network to prove our argument. To this aim, we back-project to 3D coordinates the matches in one view frame, the same utilized for our RAUM-VO, by interpolating the depth map values with the bilinear sampler of STNs.

Consequently, we can apply PnP with RANSAC to estimate the two view motions for all the sequences. Remarkably, the outcome of PnP, on average, matches closely that of the pose network (see Table 3.4), especially for the training sequences when we fix the rotation with F2F. This result aligns with those of, for example, DeepMatchVO [251] or DF-VO [314], which do not obtain significantly better odometry results by leveraging PnP directly during the training or at the test time. Interestingly, though, the combination of a PnP with the estimated depths works best for the test sequences, indicating that this approach may generalize more.

Table 3.2: Network prediction results separately mixed with ground-truth data.

The table shows an insight into the possible margins for improvement in the pose predictions coming from unsupervised methods. Hence, we substitute the ground-truth translations and rotations alternately in the pose network estimates. We show the variation in the relevant metrics for the KITTI test sequences 9 and 10.

	Metrics	**09**	**10**
Simple-Mono-VO	t_{err}	13.625	11.131
	r_{err}	3.146	4.784
	ATE	66.591	18.792
	RPE (m)	0.166	0.077
	RPE (°)	0.067	0.083
Ground-Truth Translation	t_{err}	13.325	11.409
	r_{err}	3.146	4.784
	ATE	65.081	20.715
	RPE (m)	0.162	0.028
	RPE (°)	0.067	0.083
Ground-Truth Rotation	t_{err}	3.029	6.038
	r_{err}	0.010	0.014
	ATE	9.026	12.894
	RPE (m)	0.070	0.080
	RPE (°)	0.005	0.005

Table 3.3: F2F solver initialization

Comparison of different initialization approaches for the LM scheme that solves the frame-to-frame motion. Overall, the rotation from the pose network is the best, followed by a constant motion model.

Initialization	**Metrics**	00	01	02	03	04	05	06	07	08	09	10	**Avg.Train**	**Avg.All**
Identity	t_{err}	6.192	8.023	5.888	3.919	2.860	7.659	9.100	10.969	5.402	3.851	9.475	6.668	6.667
	r_{err}	2.222	1.025	1.670	1.909	0.645	3.340	2.926	6.565	1.926	0.742	2.605	2.470	2.325
	ATE	39.195	21.231	91.621	2.651	2.283	40.192	19.682	20.592	30.142	12.939	13.399	29.732	26.721
	RPE (m)	0.040	0.259	0.060	0.030	0.052	0.039	0.046	0.036	0.052	0.069	0.077	0.068	0.069
	RPE (°)	0.100	0.101	0.072	0.082	0.035	0.083	0.059	0.158	0.067	0.070	0.088	0.084	0.083
Constant Motion	t_{err}	6.062	12.009	5.823	6.606	2.860	5.877	3.033	2.481	19.533	3.255	5.843	7.143	6.671
	r_{err}	2.128	1.833	1.728	3.119	0.645	2.105	0.837	1.150	7.772	0.862	0.683	2.368	2.078
	ATE	58.308	49.099	79.710	6.678	2.283	29.920	9.234	2.258	99.024	11.190	12.297	37.390	32.727
	RPE (m)	0.044	0.265	0.056	0.030	0.052	0.039	0.046	0.028	0.160	0.069	0.078	0.080	0.079
	RPE (°)	0.075	0.086	0.059	0.066	0.035	0.060	0.042	0.068	0.702	0.072	0.051	0.133	0.120
Pose Network (RAUM-VO)	t_{err}	2.548	8.354	2.578	3.217	2.860	3.045	3.033	2.390	3.632	2.927	5.843	3.517	3.675
	r_{err}	0.775	0.868	0.582	1.334	0.645	1.153	0.837	1.037	1.074	0.318	0.683	0.923	0.846
	ATE	16.272	23.748	16.139	2.602	2.283	17.470	9.234	2.164	16.303	8.664	12.297	11.802	11.561
	RPE (m)	0.040	0.257	0.050	0.030	0.052	0.038	0.046	0.028	0.053	0.068	0.078	0.066	0.067
	RPE (°)	0.059	0.062	0.048	0.048	0.035	0.044	0.042	0.058	0.045	0.042	0.051	0.049	0.049

Table 3.4: PnP vs pose network

Comparison of the trajectory estimated by PnP combined with the depth network and the poses predicted by our trained network.

Poses Source	**Metrics**	00	01	02	03	04	05	06	07	08	09	10	**Avg.Train**	**Avg.All**
Pose Network (Simple-Mono-VO)	t_{err}	9.365	8.920	6.830	3.697	2.570	4.964	3.138	3.568	7.125	13.625	11.131	5.575	6.812
	r_{err}	2.840	0.562	1.582	2.478	0.566	2.083	0.959	1.866	2.608	3.146	4.784	1.727	2.134
	ATE	94.949	30.004	83.155	4.112	2.377	30.227	8.726	8.872	59.887	66.591	18.792	35.812	37.063
	RPE (m)	0.090	0.304	0.087	0.037	0.055	0.041	0.051	0.044	0.074	0.166	0.077	0.087	0.093
	RPE (°)	0.072	0.042	0.057	0.048	0.036	0.049	0.040	0.048	0.052	0.067	0.083	0.049	0.054
PnP	t_{err}	6.808	17.627	6.319	4.046	2.627	4.629	2.981	3.013	6.360	7.019	6.708	6.045	6.194
	r_{err}	2.190	1.195	1.339	2.364	0.582	1.863	0.781	1.691	2.317	2.029	2.644	1.591	1.727
	ATE	79.125	63.596	76.800	4.402	2.424	29.000	8.660	7.106	52.700	35.664	9.576	35.979	33.550
	RPE (m)	0.061	0.636	0.086	0.033	0.055	0.039	0.049	0.040	0.067	0.082	0.073	0.118	0.111
	RPE (°)	0.060	0.057	0.049	0.042	0.029	0.039	0.032	0.036	0.043	0.068	0.085	0.043	0.049
F2F rotation w/ PnP translation	t_{err}	2.796	15.552	2.775	3.482	3.123	3.008	3.164	2.373	3.876	3.072	4.343	4.461	4.324
	r_{err}	0.775	0.868	0.582	1.334	0.645	1.146	0.837	0.861	1.074	0.318	0.683	0.902	0.829
	ATE	17.662	41.782	15.194	2.342	2.459	17.203	9.451	3.983	16.741	8.288	8.909	14.091	13.092
	RPE (m)	0.043	0.527	0.053	0.034	0.055	0.040	0.050	0.035	0.055	0.071	0.073	0.099	0.094
	RPE (°)	0.059	0.062	0.048	0.048	0.035	0.045	0.042	0.059	0.046	0.042	0.051	0.049	0.049
F2F rotation w/ Pose Network translation (RAUM-VO w/o L_r)	t_{err}	2.829	9.870	2.766	4.146	3.080	3.029	3.177	2.802	3.804	3.130	5.875	3.945	4.046
	r_{err}	0.775	0.868	0.582	1.334	0.645	1.146	0.837	0.861	1.074	0.318	0.683	0.902	0.829
	ATE	18.339	28.499	15.497	2.468	2.419	17.363	9.502	4.732	16.426	9.033	12.410	12.805	12.426
	RPE (m)	0.043	0.307	0.053	0.037	0.055	0.041	0.051	0.036	0.056	0.070	0.079	0.075	0.075
	RPE (°)	0.059	0.062	0.048	0.048	0.035	0.045	0.042	0.059	0.046	0.042	0.051	0.049	0.049

3.6.2 Comparison with DF-VO

We can probably ascribe the success of DF-VO to an accurately trained OF, which provides a significantly higher number of precise matches in thousands. Still, these correspondences are specific to the scenario they use to train the OF network. Conversely, the 2D features detected by Superpoint are fast to compute, distinctly identified, repeatable, and, more importantly, sparser (a few hundred). Therefore, we note that the OF network can hardly reach the generalization capability of a dedicated feature extraction network. Additionally, due to dense but noisy correspondences, DF-VO needs to iteratively search the best fit mode (*e.g.*, based on the number of inliers), and decide between the essential or homography motion model with multiple RANSAC routines. While this approach accurately describes the two-view motion of the KITTI sequences, it is computationally expensive. Instead, RAUM-VO uses all the matches found by Superglue for solving the eigenvalue minimization problem of F2F only once, adding minimal overhead to the pose network run-time. Thus, we remove the need for repeated samples of the correspondences and avoid the numerous estimation of homography and essential matrices with the related model selection strategy. Therefore, we resort to the output of the pose network and a single model-free rotation adjustment step, which is comparably a more efficient approach.

Furthermore, another potential determining factor of success is the depth scale consistency. DF-VO considers the depth maps as a source of multiple hypotheses for the translation vector scale. Thus, we can presume that the disparities jointly learned with the OF have a higher degree of long-term scale consistency and structure accuracy. In this way, the DF-VO scale alignment procedure can recover the best norm for the translation vector, which the employed Nister 5-point [207] algorithm delivers only up to a scale factor. In addition, the depth consistency loss may not be as effective as the consistency loss between rigid motion and OF in maintaining a unique long-term scale factor.

Consequently, for evaluating our depth scale consistency, we applied a scale alignment procedure similar to DF-VO for scaling the translation solutions obtained from the F2F and essential matrix, using the implementations of OpenGV [143] and OpenCV, respectively. First, we pick the essential matrix with the most inliers after ten iterations, sampling 20% of the matches each time and estimating it using RANSAC with threshold 10^{-3}. Next, we triangulate the 2D correspondences and keep only those that pass the cheirality check. Finally, we sample 80% of the triangulated points $\mathbf{X}_t$ ten times and fit a linear model with RANSAC:

$$\mathbf{Y}_d = s\,\mathbf{X}_t \tag{3.18}$$

to find the coefficient s that maps $\mathbf{X}_t$ to $\mathbf{Y}_d$, which is the set of 3D points obtained by projecting the matches with the estimated depths. Finally, we take the scale s that has the minimum $\delta = ||1 - s||_2$. We fall back to the pose-network-estimated translation only if less than 51% of matches do not pass the cheirality check or if $\delta > 5 \times 10^{-1}$. We accept the F2F or essential matrix translation in 93–97% of the cases with these loose constraints. We present the result of this test in Table 3.5. Still, we could not obtain a better translation than the pose network's output. Besides, the multiple RANSAC routines and sampling matches from dense correspondences may grant a decisive advantage to DF-VO. We leave a deeper analysis to understand the factors at stake for future works.

Table 3.5: Scale alignment.

Results of the scale alignment procedure applied to the translation vector from the F2F and the essential matrix estimated motions.

	Metrics	**09**	**10**
F2F Translation	t_{err}	4.14	5.68
	ATE	12.91	11.67
	RPE (m)	0.114	0.091
Essential Matrix Translation	t_{err}	4.02	5.99
	ATE	11.77	12.42
	RPE (m)	0.124	0.099
Pose Network (RAUM-VO)	t_{err}	2.927	5.843
	ATE	8.664	12.297
	RPE (m)	0.068	0.078

3.7 Summary

In this chapter, an unsupervised learning method for ego-motion estimation has been presented. The approach, named RAUM-VO, combines the translation predicted by a pose network with the rotations estimated by a geometrical method called F2F. Also, an additional self-supervised loss has been introduced to guide the training. More importantly, during online inference, the rotations predicted by the pose network are adjusted with a single estimation of F2F, avoiding complex strategies for model selection and multiple RANSAC loops. In addition, RAUM-VO uses Superpoint with Superglue to find robust 2D correspondences in place of randomly sampling OF, thus reducing training time and generalizing to more environments. Finally, RAUM-VO has been evaluated on the KITTI odometry dataset and compared with other relevant state-of-the-art methods. Remarkably, this adjustment step is decisive for improving the prediction of unsupervised pose networks while being more efficient than using OF and essential matrix estimation.

4
Supervised Global Pose Estimation

In this chapter [*], we address the problem of global metric localisation, using an efficient CNN architecture as a feature extractor and an MLP to regress the pose vector. Thus, we leverage MobileNetV2 [238] as a network backbone to process the images, allowing us to achieve a trade-off between competitive performances and computation speeds.

Precise and robust localisation is of fundamental importance for robots required to carry out autonomous tasks. Above all, in the case of UAVs, efficiency and reliability are critical in developing solutions for localisation due to the limited computational capabilities, payload and power constraints. On the other hand, GNSS is a standard solution to the problem of retrieving a global position. Still, it often fails due to signal loss in cluttered environments like urban canyons or natural valleys. Moreover, the precision of GNSS correlates with the number of satellites in direct line of sight [8]. Significantly, the accuracy requirements for robotics applications are often not met by some technological implementations, such as the GPS, where the provided measurements can be affected by uncertainty up to some meters. As an alternative to GPS, visual based localization (VBL) [216] refers to the set of methods that estimate the 6 DoF pose of a camera, that is, a composition of translation and rotation, in a global coordinate system defined at the centre of the map of the navigation environment, solely relying on the information enclosed in the images. In robotics, VBL is related to solving the kidnapped robot problem, *i.e.*, locating a robot without knowledge of the previous location and its movements. Whereas, in a SLAM pipeline, VBL is part of a relocalisation module that allows recovering the global position in the map and is applied, for example, after losing the feature tracking, for loop-closing in PGO, or for reducing the drift of the odometry estimation [269].

Visual localisation methods can be categorised either as indirect, also called topological or appearance-based, or direct, sometimes referred to as metric [216]. On the one hand, indirect VBL formulates the localisation as an image-retrieval problem, providing a rough estimate of the current location resulting from a database search for the most similar place. Nevertheless, the precision depends on the granularity of the space represented by the images saved in the database with a known associated pose [52, 302]. Also, these methods are generally inserted

[*]The work presented in this chapter has been published in the proceedings of the *International Conference on Computer Analysis of Images and Patterns* [50] and of the *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)* [49], and reproduced herein with few modifications.

among the loop closure techniques since they aim only to recognise a previously visited place and delay to a later stage the recovery of the relative pose between the loop-connected pose nodes (see section 1.2). On the other hand, direct methods cast localisation as a regression problem and thus provide both position and orientation for each new camera view without other processing steps. Therefore, the direct localisation approach is the most appropriate when the operating environment is confined to a well-defined area. We expect to obtain a pose as precise as possible when the tracking is lost.

Finally, to achieve real-time run-times and low latency predictions, we leverage an efficient deep neural architecture for the problem of 6 DoF pose estimation from single RGB camera images. In particular, we exploit the MobileNetV2 design to jointly regress the position and orientation of the camera relative to the navigation environment. Experimental results show that the proposed network can retain similar outcomes concerning the most popular SotA methods while being smaller and with lower latency, which are fundamental aspects for real-time robotics applications.

4.1 Literature Review

This section reviews the methods proposed in the literature on VBL techniques. Currently, the approaches to direct localisation follow three distinct directions. The first relies on establishing matches between 2D image features and 3D points of a structured environment model usually generated by SfM [247, 305]. Another general approach is to exploit machine learning algorithms to learn the 3D world coordinates associated with each image pixel to estimate the transformation from the camera's system of coordinates to the world-centred coordinate frame. Lastly, it is possible to obtain a solution to the global localisation by training "end-to-end" a neural network, *e.g.*, a CNN, for predicting the most probable 6 DoF directly pose associated with an input image.

Structure-based localisation indicates the family of methods supported by a 3D reconstruction of the environment created through an SfM pipeline [247] offline or with other mapping techniques that can guarantee high global consistency and accuracy of the locations of the 3D points. Hence, these methods can establish correspondences between local 2D features extracted from a query image and those associated with the 3D points in the model. Finally, the PnP algorithm [131, 160] can recover the camera's pose using a putative matches set's sample to iteratively search for the lowest reprojection error solution with a RANSAC robust fitting scheme [221]. Since the points in the 3D map Irschara *et al.*. [126] use image retrieval methods in conjunction with a compressed scene representation composed of natural and synthetic views. Li *et al.*. [168] propose to invert the search direction using a prioritisation scheme. Sattler *et al.*. [241] enhance the 2D-3D matching with a Vocabulary-based Prioritised Search (VPS) that estimates the matching cost for each feature to improve the performances and combines the two opposite search directions [242]. Despite being precise when correct correspondences are found, the main drawbacks are the computational costs, which do not scale with the extent of the area to cover, and the need to store a 3D model [216].

Scene coordinates regression methods use machine learning to speed up the matching phase by directly regressing the 3D scene coordinates associated with the image pixels. Shotton *et al.*. [252] train a random forest on red green blue and depth (RGB-D) images and formulate the localisation problem as an energy function minimisation over the possible camera location hypothesis. Hence, they use the Kabsch algorithm [132] inside a RANSAC loop to iteratively refine the hypothesis selection. The downside of these methods is the need for depth maps and high-resolution images to work well.

End-to-end learning has been adopted only recently to solve the direct localisation problem. Following the success of neural networks in many computer vision tasks ranging from image classification to object detection [153], PoseNet [137] is the first work in which CNNs are applied to the pose regression task. In particular, they reuse a pre-trained GoogLeNet [267] architecture on the ImageNet dataset [64], demonstrating the ability of the network to generalise to a completely different task thanks to transfer learning [69]. In later works, Walch *et al.*. [296] extend PoseNet with LSTM [118] to encode contextual information, and Wu *et al.*. [306] generate a synthetic pose to augment the training dataset. Subsequently, Kendall *et al.*. [135] introduce a novel formulation to remove weighting hyperparameters from the loss function. Though these single CNN methods for pose regression could not surpass the average performance of classical approaches [243, 252], they are capable of handling the most visually tricky frames, more robust to illumination variance, cluttered areas, and textureless surfaces [296].

Multi-task networks [220, 290] demonstrate that by leveraging auxiliary task learning, such as Visual Odometry or Semantic Segmentation, the neural network improves on the main task of global localisation. As a result, they were able to outperform the state-of-the-art of feature-based and scene-coordinate regression methods.

4.2 End-to-End Pose Regression

Inspired by previous works on direct visual localisation exploiting CNNs [135, 306], we aim to estimate the camera pose from a single red green blue (RGB) image by adding a regressor fed by the network's output chosen as a base feature extractor. In the following subsections, we describe the representation of the pose vector, the loss function used to learn the pose estimation task, and the architectural details of the deep learning model.

4.2.1 Pose representation

The output for each input image consists of a 7-dimensional vector **p**, representing both the translation and rotation of the camera relative to the navigation environment:

$$\mathbf{p} = [\mathbf{x}, \mathbf{q}] \tag{4.1}$$

where $\mathbf{x} \in \mathbb{R}^3$, represents the position in the 3D space and the orientation $\mathbf{q} \in \mathbb{R}^4$ is expressed as a quaternion.

Our choice of using a quaternion over other representations for the orientation is motivated by the fact that any 4-dimensional vector can be mapped to a valid rotation by scaling its norm to unit length. Instead, opting for rotation matrices would require enforcing the orthonormality constraint since the set of rotation matrices belongs to the special orthogonal Lie group, $SO(3)$ [135]. Other representations, such as Euler angles and axis-angle, suffer from the problem of periodic repetition of the angle values around 2π. Wu *et al.*. [306] proposed a variant of the Euler angles, named Euler6, to overcome the issue of periodicity in which they regress a 6-dimensional vector $e = [sin\varphi, cos\varphi, sin\theta, cos\theta, sin\psi, cos\psi]$. Notwithstanding in [306], the authors showed an improvement empirically over quaternions, we decided not to express the rotation as Euler6 for a closer comparison with the majority of the SotA approaches, unsure of whether these different representations consist of a significant improvement.

4.2.2 Loss Function

To train the network for the task of pose estimation, we minimise the difference between the ground truth pose, $[\mathbf{x}, \mathbf{q}]$, associated with an image $\mathcal{I}$ in the training dataset, and the pose predicted by the deep learning model, $[\hat{\mathbf{x}}, \hat{\mathbf{q}}]$. Hence, the loss function aims to optimise the two components of the pose, translation and orientation, denoted by $\mathcal{L}_x$ and $\mathcal{L}_q$, respectively:

$$\mathcal{L}_x(\mathcal{I}) \;=\; \|\mathbf{x} - \hat{\mathbf{x}}\|_p \,, \tag{4.2}$$

$$\mathcal{L}_q(\mathcal{I}) \;=\; \left\| \mathbf{q} - \frac{\hat{\mathbf{q}}}{\|\hat{\mathbf{q}}\|_2} \right\|_p \,, \tag{4.3}$$

where with the notation $\|\cdot\|_p$ we refer to the p-norm. In our experiments, we apply $p = 2$, which corresponds to the Euclidean norm. Besides, the predicted quaternion is normalised to unit length to ensure a valid rotation representation.

Since the two components, $\mathcal{L}_x$ and $\mathcal{L}_q$ of the loss function we want to minimise are on a different scale, a weight β is added to the quaternion error to balance the backpropagated gradient magnitudes [137]. In light of this, the complete loss function is defined as follows:

$$\mathcal{L}(\mathcal{I}) = \mathcal{L}_x(\mathcal{I}) + \beta \cdot \mathcal{L}_q(\mathcal{I}) \tag{4.4}$$

To minimise the hyperparameter from the loss function, [135] replace β with two learnable variables, $\hat{s}_x$ and $\hat{s}_q$, in the formulation of the loss with homoscedastic uncertainty:

$$\mathcal{L}(\mathcal{I}) = \mathcal{L}_x(\mathcal{I}) \cdot exp(-\hat{s}_x) + \hat{s}_x + \mathcal{L}_q(\mathcal{I}) \cdot exp(-\hat{s}_q) + \hat{s}_q \tag{4.5}$$

Homoscedastic uncertainty, contrary to heteroscedastic uncertainty, is independent of the input data and, therefore, is not an output of the model. Instead, it captures the uncertainty of the model relative to a single task, treating the regression of translation and rotation as two separate tasks while training the model for multiple tasks simultaneously. For this reason, it is helpful in multi-task settings to weigh the loss components based on the different measurement units relative to the particular task [136]. In our experiments, we initialised $\hat{s}_x$ and $\hat{s}_q$ to 0.5 and 0.1 respectively.

4.2.3 Deep Learning model

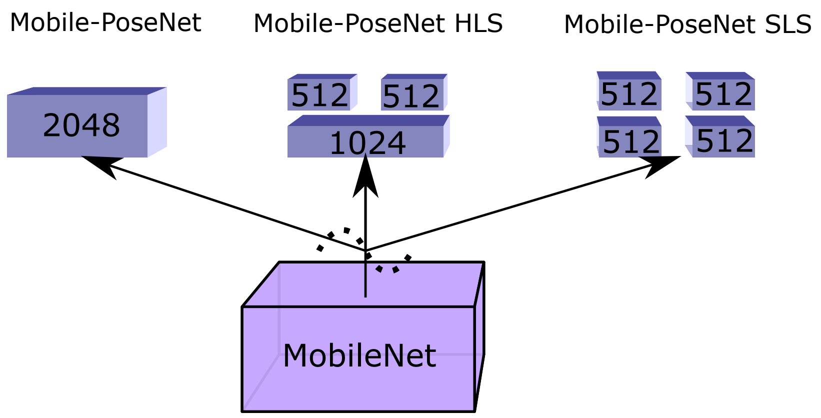


Figure 4.1: Architecture of Mobile-PoseNet

The diagram shows the overall structure of the network and how the last layers are symmetrically branched.

Since the current approaches rely on very deep network architectures, *e.g.*, GoogLeNet, we propose replacing them with a more efficient architecture to produce a more appealing solution for deployment on a UAV. Improving on the previous generation of "mobile", *e.g.*, efficient, networks [121], MobileNeV2 [238] combine the depthwise separable convolution with a linear bottleneck layer drastically decreasing the number of operations and weights involved in the computation of the output. We show that this shallower network can run faster than other single CNN solutions without sacrificing the localisation accuracy.

To build a small network for localisation, we adopted the MobileNetV2 [238] architecture by adding fulling connected layers to regress the pose. Hence, we refer to our proposed network as Mobile-PoseNet.

MobileNetV2 leverages the depthwise separable convolution to reduce the number of parameters and the multiply additions (madds) operations as in the previous iteration [121]. In addition, they introduce the linear bottleneck that reformulates the original residual block in a more efficient design[113], which supports the propagation of the gradient across numerous stacked layers. Moreover, MobileNetV2 allows tuning a width multiplier α to choose the best trade-off between accuracy and size of the network. For example, we set $\alpha = 1$ to obtain a network with 3.4M parameters and 300M madds, resulting in a sensible shrinking compared to GoogLeNet with 6.8M parameters and 1500M madds.

Thus, we perform an average pooling on the output of MobileNetV2's last convolutional layer, deriving a vector of 1×1280 dimension that contains a high-dimensional feature representation of the input image. Therefore, we connect a fully connected layer of 2048 neurons followed by a ReLu6 [147] non-linearity, which maps the features to the desired 7-dimensional pose vector. Krizhevsky *et al.*. argue that the ReLu6 helps to learn a sparse feature representation earlier in training than the ReLU counterpart. More importantly, it can exploit fixed-point low-precision calculations [121] on supported hardware.

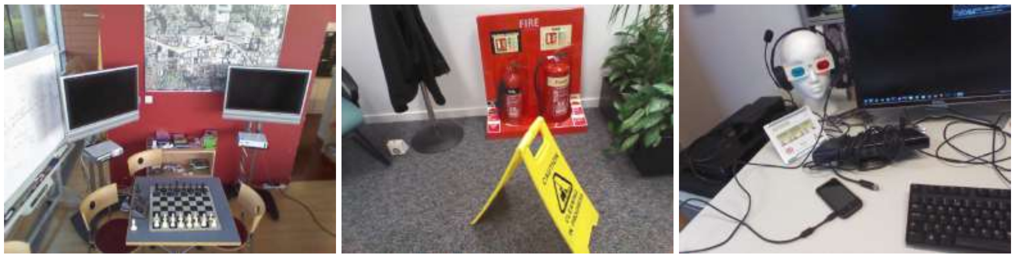
Furthermore, to improve the generalisation capability of the network, we add a batch normalisation layer [124] before the non-linearity. In addition, we adopt Dropout [260], which, as described in chapter 2, is an alternative form of activation regularisation that reduces overfitting and indirectly induces sparsity by dropping random neurons at training time.

Ultimately, inspired by the branching technique proposed in [306], we offer to regress the translation and rotation vectors separately (see Figure 4.1) by designing two other versions of the pose regressor MLP (see Figure 4.1). Hence, we symmetrically split the neurons into two groups of 1024 to maintain the same total number intact. Additionally, we experiment with a third version of the network that keeps a shared fully connected layer for translation and rotation of 1024 neurons and splits in half the rest forming two groups of 512. Our purpose is to compare the benefits of jointly learning position and orientation, that is, sharing the information enclosed in the shared weights, against training two individual branches for each task. Therefore, we distinguish these design choices by referring to the first as symmetric layer split (SLS), and the latter as half layer split (HLS).

4.3 Experiments and Results

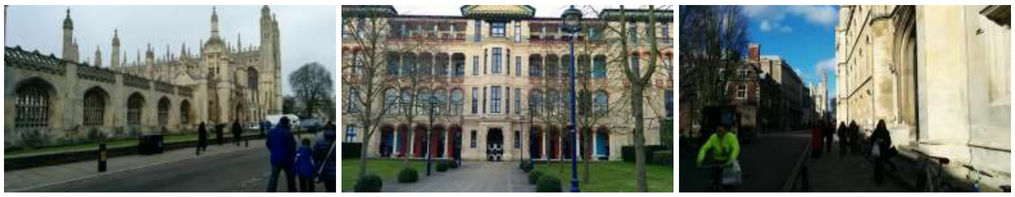
In this section, we evaluate our trained models on 7-Scenes [252] and Cambridge Landmarks datasets [137]. The first includes indoor images, whereas the second one contains pictures captured in an outdoor urban environment. These datasets have been adopted by most of the relevant SotA methods. Using an indoor dataset also allows the analysis of how the proposed method behaves in scenarios with characteristics different from outdoor urban settings.

4.3.1 Datasets



(a) Chess (b) Fire (c) Heads

Figure 4.2: 7-Scenes sample images



(a) King's College (b) Old Hospital (c) Street

Figure 4.3: Cambridge Landmarks sample images

7-Scenes [252] is a dataset for RGB-D designed to benchmark relocalisation methods. Thus, it was collected through a Microsoft Kinect camera in seven indoor scenarios, which contains more than 40K frames with 640×480 resolution and an associated depth map. This dataset's challenging aspects are its high camera pose variations in a small area generating motion blur, perceptual aliasing, and light reflections. These unique characteristics make the pose estimation particularly difficult for methods relying on handcrafted features, especially in views where textured areas are not distinguishable [296].

Cambridge Landmarks was introduced in [137], and currently provides six outdoor scenarios. It contains more than 10K images sampled from a high-resolution video captured by a smartphone. The ground truth labels were generated through an SfM reconstruction of the environment. Visual clutter caused by the presence of pedestrians and vehicles plus a substantial variance in the lighting conditions is the main challenge posed by this dataset.

All the scenes in both datasets are subdivided into sequences, depending on the trajectories from which they were generated. Each of the sequences shows a different perspective of the surrounding environment. Hence, for training and testing our model, we use the same partitioning of the datasets provided by the respective authors. Thus, we create a separate validation (val) set for evaluating the models during the training phase by taking a random sample of 10% of the frames from all the sequences in the training set. To better approximate the test set's performance more accurately, we form the val set from sequences that are not overlapping those in the training set.

4.3.2 Experimental Setup

The network is implemented using the TensorFlow open-source library [1]. We initialised MobileNet with weights pre-trained on the ImageNet dataset and the fully connected layers using the method proposed by He *et al.*. [112]. Before training, we normalise the images by computing an RGB image representing the standard deviation and the mean of a particular dataset scene. Then, we remove the mean for each image and divide it by the standard deviation to centre the data and uniformly scale the pixel intensities. The dropout rate is set to 0.1, which means only 10% of the neurons are turned off during training, whereas the batch normalisation moving average momentum is set to 0.99. We optimised the models using Adam [139] with a learning rate $\alpha = 1e^{-4}$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$, on batches of size 128 shuffled at each new epoch, using an NVIDIA Tesla V100 with 16GB of memory. Thus, we let the training last until the convergence of the loss is reached on the "dev" set.

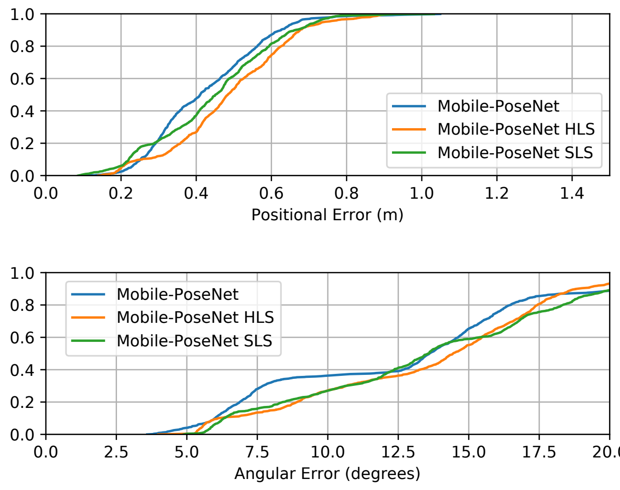
4.3.3 Discussion of the results

In Table 4.1, we compare the results with three other CNN-based localisation methods: PoseNet [137], PoseNet2 [135] with learned σ^2 weights in the loss, and BranchNet [306], which represents rotations with Euler6 and splits the network into two branches to regress the position and the orientation separately. Whereas we benchmark our result against PoseNet [137] because it pioneered the approach to the direct localisation problem using CNNs, we share with the other methods some architectural choices. On the one hand, we adopt the homoscedastic uncertainty introduced by [135] to balance different loss components; on the other hand, we split the network layers following the work of [306], who showed significant improvements.

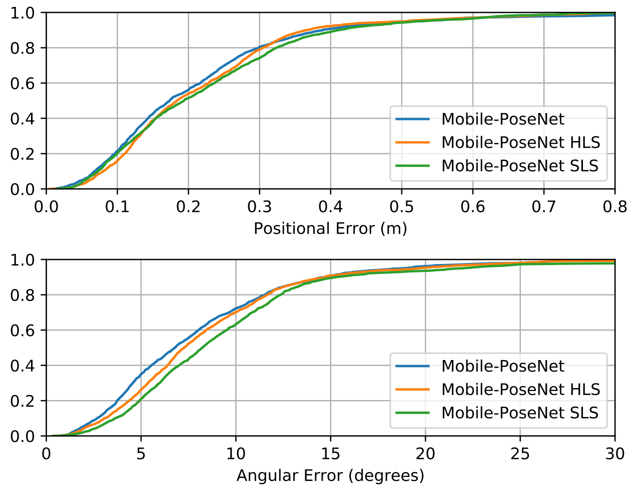
In general, Mobile-PoseNet can outperform PoseNet and BranchNet in most scenarios. The automatic balancing of the translation and rotation components of the loss function is the main factor that gives us an advantage over these methods. Instead, PoseNet2 obtains the best results in all the benchmark scenes apart from Old Hospital in which Mobile-PoseNet HLS can surpass the translation error by a small margin. Also, we note that PoseNet2 uses frames with a resolution of 256×256, whereas our models require a minor input of 224×224. Besides, we observe that Mobile-PoseNet performs better on scenes spread over smaller areas. In contrast, Mobile-PoseNet HLS and SLS competitively gain higher scores in Cambridge Landmarks scenarios with an elevated spatial extent.

Finally, we run the network on a TegraTX2 to test the latency, that is, the time interleaving from the submission of one frame into the network to the moment of receiving the estimated pose. Hence, using the integrated TensorFlow tool for run-time statistics, we note that MobileNet-PoseNet takes on average 17.5ms of run time, while the classic PoseNet 24ms.

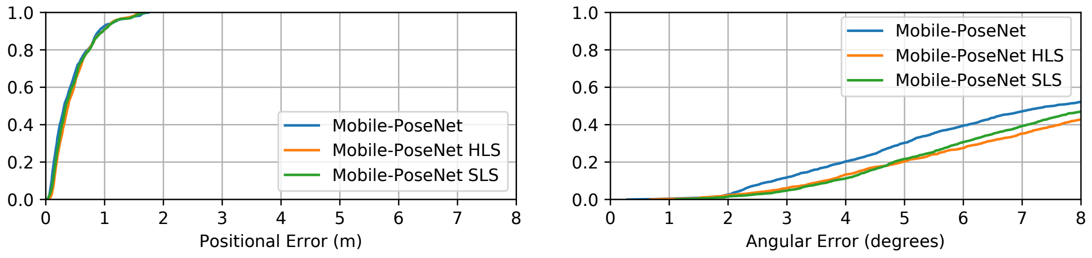
At last, we want to remark that the proposed solution employs a base feature extractor that carries half the number of parameters, in contrast to the aforementioned state-of-the-art methods we compare. This factor inevitably impacts the accuracy of the output.



(a) Stairs

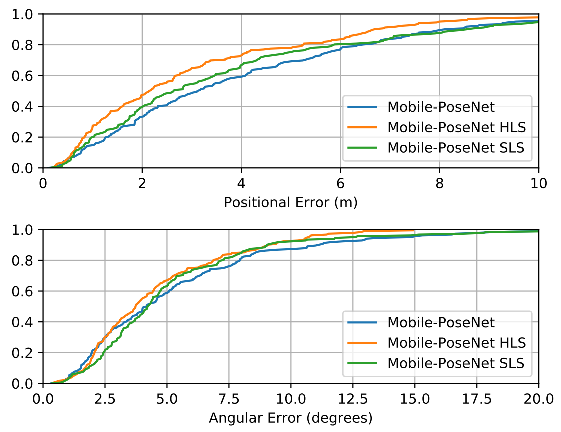


(b) Chess

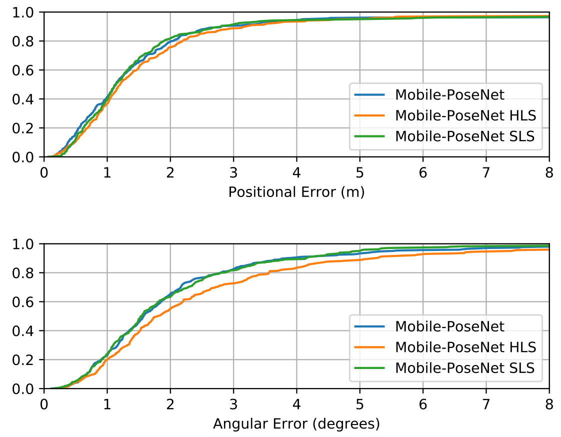


(c) Pumpkin

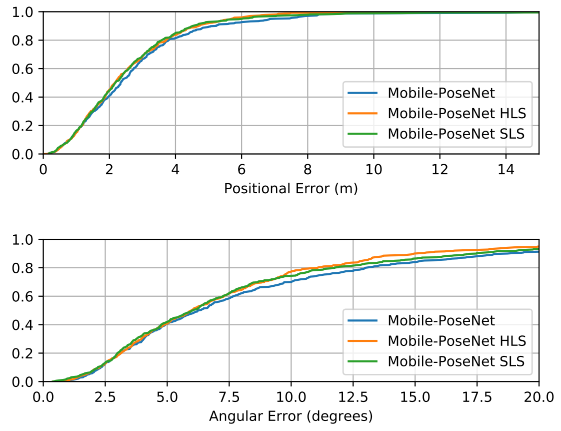
Figure 4.4: Cumulative distribution of the localisation error



(a) Old Hospital



(b) King's College



(c) St. Mary's Church

Figure 4.5: Cumulative distribution of the localisation error

Table 4.1: Median localisation errors on the 7-Scenes and Cambridge Landmarks datasets

	Area or Volume	BranchNet[306] Euler6	PoseNet[137] β Weight	PoseNet2[135] Learn σ^2 Weights	Mobile-PoseNet (proposed)	Mobile-PoseNet HLS (proposed)	Mobile-Posenet SLS (proposed)
7-Scenes							
Chess	$6m^3$	0.20m, 6.55°	0.32m, 8.12°	0.14m, 4.50°	0.17m, 6.78°	0.18m, 7.27°	0.19m, 8.22°
Fire	$2.5m^3$	0.35m, 11.7°	0.47m, 14.4°	0.27m, 11.8°	0.36m, 13.00°	0.36m, 13.6°	0.37m, 13.2°
Heads	$1m^3$	0.21m, 15.5°	0.29m, 12.00°	0.18m, 12.1°	0.19m, 15.3°	0.18m, 14.3°	0.18m, 15.5°
Office	$7.5m^3$	0.31m, 8.43°	0.48m, 7.68°	0.20m, 5.77°	0.26m, 8.50°	0.28m, 8.98°	0.27m, 8.54°
Pumpkin	$5m^3$	0.24m, 6.03°	0.47m, 8.42°	0.25m, 4.82°	0.31m, 7.53°	0.38m, 9.30°	0.34m, 8.46°
Red Kitchen	$18m^3$	0.35m, 9.50°	0.59m, 8.64°	0.24m, 5.52°	0.33m, 7.72°	0.33m, 9.19°	0.31m, 8.05°
Stairs	$7.5m^3$	0.45m, 10.9°	0.47m, 13.8°	0.37m, 10.6°	0.41m, 13.6°	0.48m, 14.4°	0.45m, 13.6°
Cambridge Landmarks							
Great Court	$8000m^2$	—	—	7.00m, 3.65°	8.68m, 6.03°	8.12m, 5.60°	8.60m, 5.58°
King's College	$5600m^2$	—	1.92m, 5.40°	0.99m, 1.06°	1.13m, 1.57°	1.20m, 1.79°	1.14m, 1.53°
Old Hospital	$2000m^2$	—	2.31m, 5.38°	2.17m, 2.94°	3.11m, 4.11°	2.13m, 3.73°	2.62m, 4.21°
Shop Façade	$875m^2$	—	1.46m, 8.08°	1.05m, 3.97°	1.39m, 6.37°	1.55m, 5.64°	1.73m, 6.19°
St. Mary's Church	$4800m^2$	—	2.65m, 8.48°	1.49m, 3.43°	2.34m, 6.23°	2.16m, 5.97°	2.18m, 6.01°
Street	$50000m^2$	—	—	20.7m, 25.7°	22.9m, 36.3°	22.6m, 32.6°	22.9m, 36.2°

4.4 Pose Estimation in Dynamic Scenes

In the event of a robot operation inside an urban environment, the localisation can be affected by the occlusions caused by dynamic objects. In some cases, where the occluding objects severely obstruct the field of view, we cannot expect any algorithm based on visual input to obtain any helpful information on the current position. Instead, we argue that if points of interest are included in some patches of the images, then a robust algorithm must provide the estimated pose based solely on these important clues without being distracted by the features of the dynamic objects. Hence, our objective is to empirically show the effectiveness of convolutional neural networks in focusing on the part of the image containing the relevant information. To confirm such a proposition, our strategy is to detect the parts of the image belonging to moving objects and to train a pose regressor using the pre-processed masked dataset. Eventually, we compare the results with those obtained by training a network on the typical dataset to test if the model outcome is sensitive to the missing features in the masked objects. Thus, we perform an ablation study where the single varying element is the input dataset for training. To this aim, during the experimental phase, we keep all the hyperparameters of the networks unchanged.

4.4.1 Object Segmentation in SLAM

Works to deal with segmented dynamics objects in the SLAM have been proposed in the past. In [301], a 3D object tracker is used to prevent a SLAM algorithm from relying on features belonging to moving parts in its map and removing features occluded by moving objects. Riazuelo *et al.*. [229] introduced a human tracker to remove certain regions from the SLAM pipeline, showing that such a strategy improves the performance of camera tracking and relocation. A solution that addresses the same problem for both movable and moving objects has been proposed by Bescos *et al.*. [15]. Authors employ a CNN to segment dynamic objects in the images to avoid extracting features made by SLAM algorithms on those parts. They also propose a reconstruction of occluded parts, but for the RGB-D case only. Mask R-CNN [111] is used to segment dynamic objects' shapes, while RGB-D information strengthens the segmentation and labels moving objects not detected from the CNN.

Instead, in [300], the object detection network YOLOv3 [226] has been used to propose a semantic SLAM in real time.

If the benefits of prior knowledge and of adding a segmentation step have been shown in classic SLAM scenarios, very few works have been provided for the case of neural networks. In fact, state-of-the-art architectures automatically learn to extract the relevant information, *i.e.*, the "important" parts of an image for the different tasks under consideration.An attempt in the SotA between masking parts of the images and neural network performance has been provided in [67], where random region masking has been used to get regularisation on the input layer (cutout). Furthermore, saliency maps can visually offer empirical evidence of the ability of neural networks to recognise relevant data. In [256], a milestone work has been proposed to visualise models for the image classification tasks of CNNs through the visual saliency maps, a topographical representation of unique features in visual processing. Furthermore, [257, 266] advanced in the methods for quantifying the input pixels' contribution to the final prediction. Hence, they propose different techniques to visualise saliency maps describing the magnitude of the backpropagated gradient. Finally, Sundararajan *et al.*. [266] introduce two axiomatic principles that saliency methods should enforce to be reliable in their evaluation. In [257], a smoothing technique is proposed in which Gaussian noise is added to the input, creating multiple intermediate saliency maps that are averaged together.

4.4.2 Dataset pre-processing

For the following experiments, we use the King's College scenario since this representation of an urban scenario is ideal for our study case due to the presence of pedestrians and vehicles (see Figure 4.6).

The phase of moving object segmentation is performed offline, pre-processing all the images in the dataset in advance. We used a pre-trained Mask R-CNN [111] (implemented by [2]) to segment automatically the objects (see Figure 4.7a). The output of Mask R-CNN is the list of object classes that are detected in the image and a mask labeling the pixels that belong to each object. From the original 90 MS COCO [174] categories, and we picked those that best fit the concept of a moving object not relevant to the localisation objective. Among those, we include things that a person could carry, *e.g.*, a backpack, and all the animals inde-

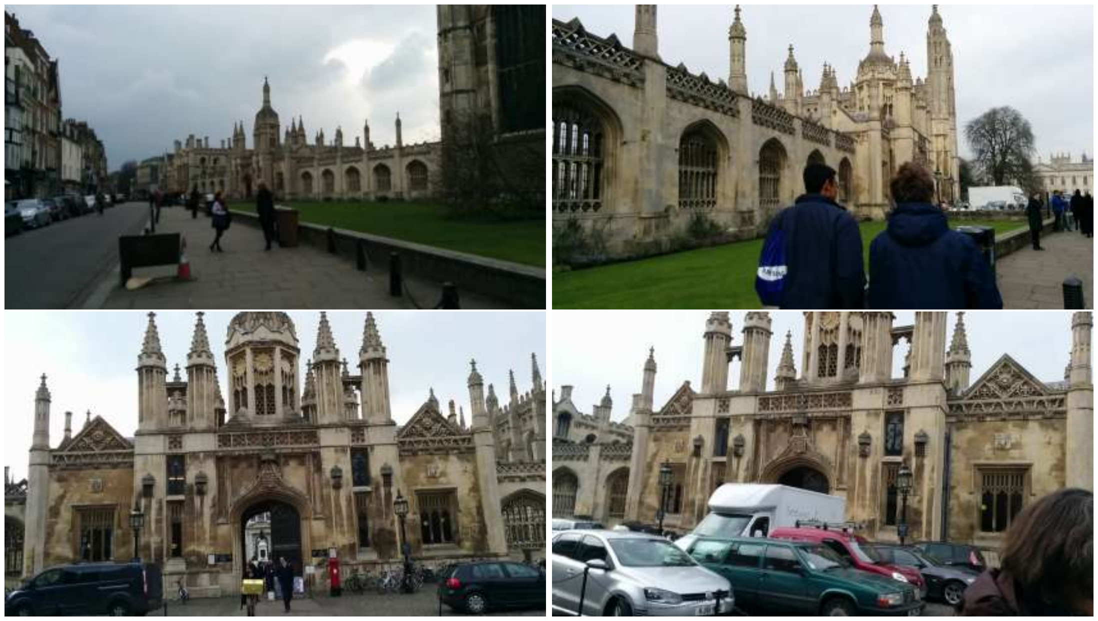


Figure 4.6: King's College sample pictures from the Cambridge Landmarks dataset

pendently from their presence in the King's college dataset, as resumed in Table 4.2.

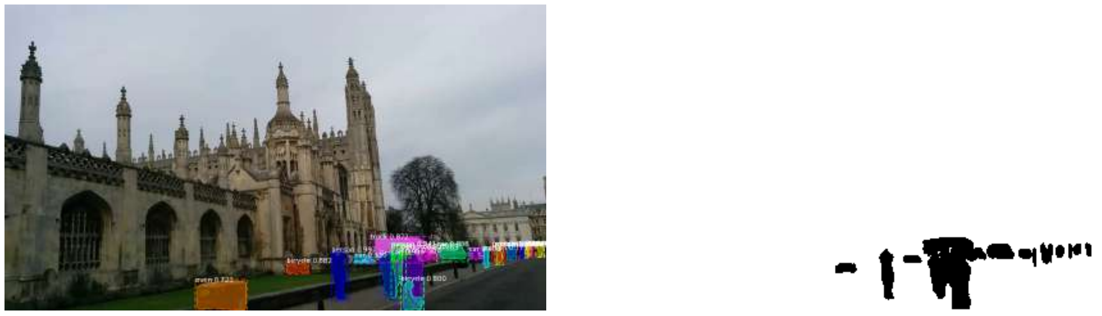
Table 4.2: Number of objects detected per each category

The object categories and the number of detected objects found for each are displayed here. Only the categories with at least one detected object are included in the table.

Category	Person	Bicycle	Car	Truck	Handbag	Backpack	Motorcycle	Suitcase
Count	6059	2976	2716	256	248	121	93	41
Category	Umbrella	Tie	Boat	Bird	Bus	Airplane	Dog	Horse
Count	9	7	5	2	2	1	1	1

Hence, the outcome of this step is a binary mask of the same size as the processed images representing whether or not the corresponding pixel has to be ignored (see Figure 4.7b).

Following, we compute the per-channel mean and standard deviation of the pixel value on the entire dataset. Before training, we standardise the images by subtracting the mean and dividing by the standard deviation to obtain zero-centred and unitary variance input distribution. Henceforth, we apply the binary mask setting to zero the input underlying the



(a) Object Segmentation (b) Binary Mask

Figure 4.7: Pre-processing applied to a sample image of the King's College sequence

black part of the mask (see Figure 4.7b) and leave unchanged the rest. This procedure retraces the cutout regularisation technique [67] in applying a zero mask after normalising the input. Ultimately, we trained using random crops of size 224×224 of the original images. Instead, during the tests, only the central crop is used.

4.4.3 Experiment Results and Discussion

Herein, we propose the experiment on global localisation using ResNetV2 [115] with 152 layers and GoogLeNet [267] as feature extractors. Then, we discuss the results obtained with ResNet and GoogLeNet and perform an ablation study on the use of the masking procedure either at training time, at test time, or during both phases.

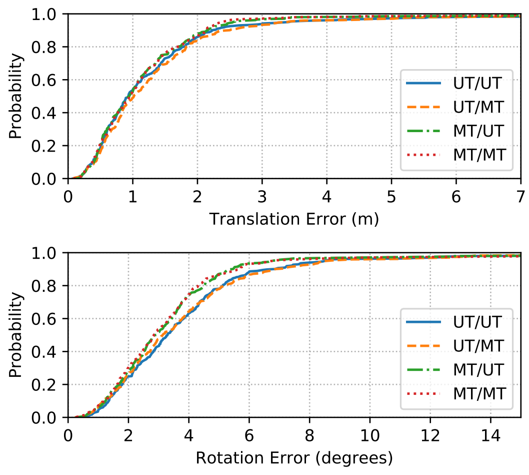
The CNNs' implementations are provided within the TensorFlow-Slim open-source library [1] and the weights are initialised from models pre-trained on the ImageNet dataset. The fully connected layers that regress the pose are initialised randomly using the method proposed by He *et al.*. [112]. The dropout rate is set to 0.12 so that 12% of the neurons are turned off during training on average, whereas the Batch Normalisation momentum is set to 0.99. Adam [139] optimiser is used to minimise the loss with a learning rate $\alpha = 1e^{-4}$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$, on batches of size 64 shuffled at each new epoch. A single NVIDIA Tesla V100 has been used for each training.

Table 4.3: Median and mean errors w/ and w/o object masking

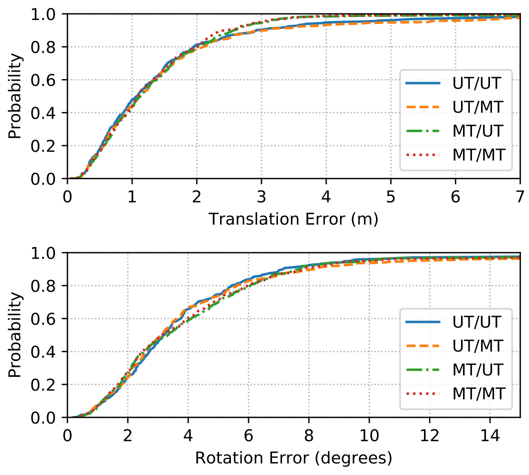
Median and mean errors of translation and rotation for the four different combinations of masking and unmasking images at training and test time.

	UT/UT	UT/UM	MT/UT	MT/MT
GoogLeNet				
Median Error	0.93m, 3.29°	1.01m, 3.12°	0.94m, 2.86°	0.95m, 2.84°
Mean Error	1.36m, 3.85°	1.39m, 3.84°	1.21m, 3.40°	1.19m, 3.39°
ResNetV2 152				
Median Error	1.06m, 3.18°	1.10m, 3.12°	1.13m, 3.31°	1.09m, 3.12°
Mean Error	1.58m, 3.99°	1.73m, 4.20°	1.40m, 4.21°	1.33m, 4.20°

Hence, we obtained the mean and median error statistics for four different combinations that we name: Unmasked Training + Unmasked Test (UT/UT), Unmasked Training + Masked Test (UT/MT), Masked Training + Unmasked Test (MT/UT), and Masked Training + Masked Test (MT/MT).



(a) GoogLeNet localisation Performance



(b) ResNetV2 152 localisation Performance

Figure 4.8: Cumulative distribution plot of the translation and rotation error

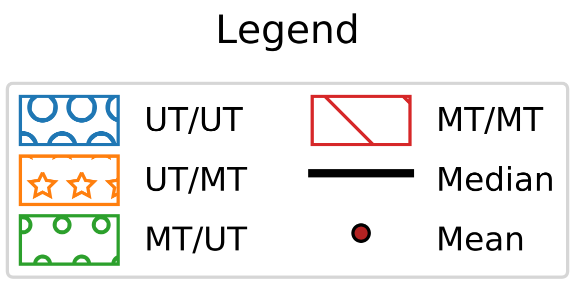
Comparing the four approaches, we can notice that the trends are overall similar.

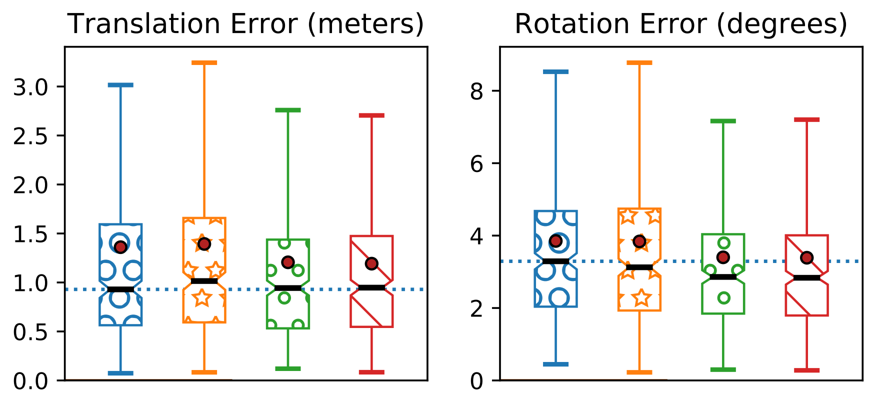
Comparing the mean and median values in Table 4.3, it is possible to observe that a method that outperforms the others does not clearly emerge. For example, while the median translation error is slightly lower in the UT/UT approach, it is the contrary for the rotation. This result is reflected in the lower plot of Figure 4.8a, in which the MT/MT (or MT/UT) approach exhibits a higher probability of obtaining lower rotation error with GoogLeNet, whereas with ResNet (upper plot in Figure 4.8b) it shows a mildly lower translation error for the last percentile of frames.

Inspecting the box plots in Figure 4.9, we notice that the medians of MT/MT fall in the 95% confidence intervals of the UT/UT respective medians (shown through the notches and a blue dotted line), apart from the rotation error of GoogLeNet. Therefore, we can conclude that the medians do not differ with 95% confidence. From one point of view, this evidence could imply that the UT/UT approach already incorporates the capability of masking irrelevant information contained in the input images. However, on the other side, it validates the prior assumption that features contained in the dynamic objects are not influencing the pose estimation and can be hidden without harming the accuracy of the results.

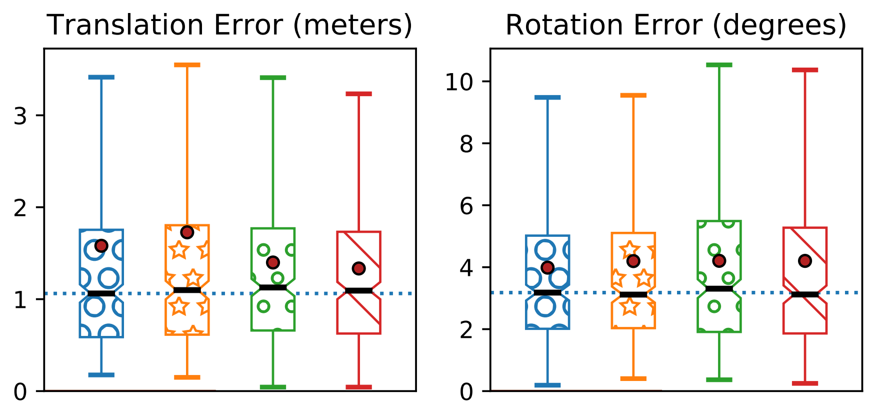
Ultimately, we study possible relationships between the portion of the image that can be masked and the error in the localisation. With this regard, we bin the test images by the percentage of pixels belonging to detected dynamic objects over the total number of pixels, *i.e.*, 224×224, which is displayed in Figure 4.11. Since the majority of images have lower than 5% masked pixels and very few over 35%, in Figure 4.10 we show the results for the bins: 0% to 5% masked pixels, which contains 207 images; 5% to 15%, containing 56 images; 15% to 35% containing 28 images.

Thus, the box plot in Figure 4.10 reveals an apparent connection between the increase in the localisation error, both translation and rotation part, and the portion of the image that is covered by dynamic objects especially when this is a significant part, *e.g.*, more than 15%. Furthermore, in the test set, there are 3 more images over the 35% threshold. These are not included in the plot since their mean error is markedly higher than the other bins' means and would not make possible a clear visualisation. Anyway, this evidence further confirms a relationship between localisation error and the size of dynamic objects.



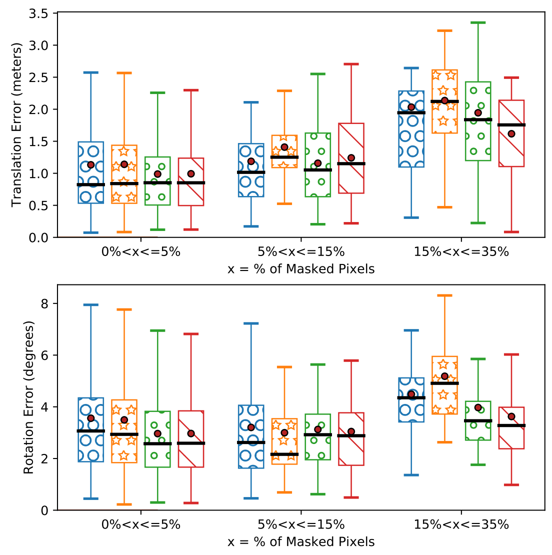


(a) GoogLeNet errors box plot

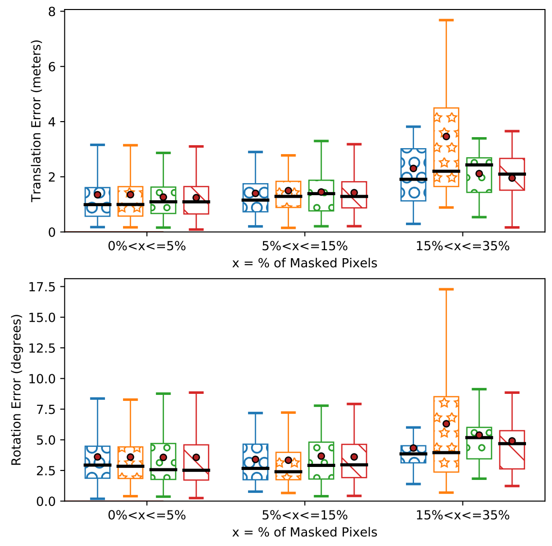


(b) ResNetV2 152 errors box plot

Figure 4.9: Box plots of the translation and rotation errors



(a) GoogLeNet errors box plot



(b) ResNetV2 152 errors box plot

Figure 4.10: Translation and rotation errors grouped by occluded pixels percentage

Translation and rotation errors on the test images are grouped by the percentage of pixels that could be masked with the proposed method (for the legend see Figure 4.9). The figure shows a slight relationship between the portion of the image that is obscured and the increase in the mean/median error.

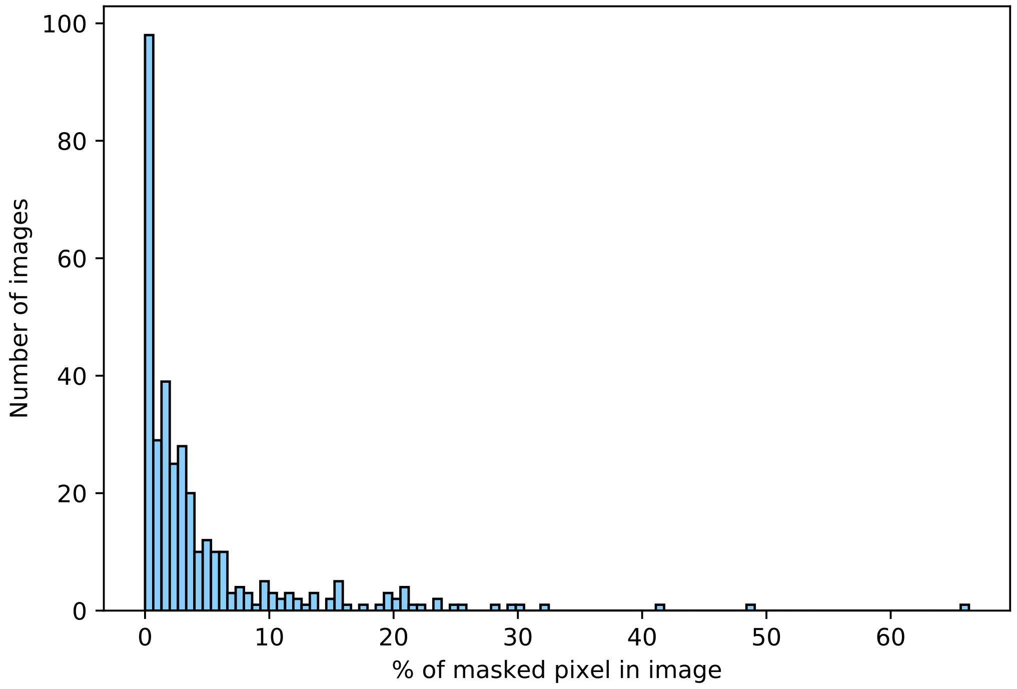


Figure 4.11: Histogram of the percentage of the masked pixels

Histogram of the distribution of the images with the percentage *x*% of masked pixels in the test set of the Cambridge King's College dataset. Among the 343 images used for testing, almost 100 have less than 1%. Instead, the remaining images present more evidence of masked objects.

4.4.4 Saliency Maps Visualisation

Herein, we investigate the contribution that each pixel is supposed to give to the final pose estimate. For this purpose, we use the saliency maps produced by SmoothGrad [257] technique combined with the Integrated Gradients method [266]. Integrated Gradients (IG) accumulates the contribution given by the pixels in the images that lie in the straight interpolation line between the original image, *e.g.*, the one for which we would like to visualise the saliency, and a baseline image, *e.g.*, a black picture which is supposed to have a neutral pose estimation (high error). Hence, it integrates the gradient of the network output with respect to each input image by computing the Riemman approximation of the integral, with a discrete number of steps m. Naming x the original image and x' the baseline, and calling F the function represented by the neural network, we calculate the saliency for the pixel i as:

$$IG_i(x) = (x_i - x'_i) \times \sum_{k=1}^{m} \frac{\partial F(x'_i + \frac{k}{m} \times (x_i - x'_i))}{\partial x_i} \times \frac{1}{m} \tag{4.6}$$

In our experiments, we use $m = 40$ integration steps.

Furthermore, SmoothGrad (SG) sharpens the saliency maps by taking into account the possible fluctuations of the backpropagated gradients. The authors showed that the gradient is sensitive to slight input variations. Henceforth, they proposed to smooth the maps by averaging the backpropagated gradient of multiple input instances created by applying a Gaussian filter. For this reason, SmoothGrad is compatible with any saliency algorithm since by itself does not compute the maps. The computation takes a saliency function S applied to an image x. Then, it iterates n times sampling additive noise from a Gaussian normal distribution $\mathcal{N}(0, \sigma^2)$ with zero mean and standard deviation σ:

$$SG(x) = \frac{1}{n} \times \sum_{k=1}^{n} S(x + \mathcal{N}(0, \sigma^2)) \tag{4.7}$$

In our experiments, we parametrised SG with $n = 35$ and $\sigma = 0.1 \cdot (x_{max} - x_{min})$, *i.e.*, the 10% of the pixel intensity range.

Finally, the saliency maps obtained using GoogLeNet for the UT/UT and MT/MT approaches are displayed for two specific scenarios: first, an object in the foreground, Figure 4.12;

second, objects in the background, Figure 4.13. It is possible to observe that in the first case, the UT/UT approach almost masks most of the cyclist shape as well as the MT/MT approach. On the contrary, the second frame shows that the gradient "leaks" inside the form of the white camper, making it visible. The vehicle's presence possibly explains this effect in the same spot in many frames of the dataset; therefore, the network overfits the features carried by its visual appearance. When the dataset presents a recurring fixed dynamic object, it may be beneficial to obfuscate those objects by masking them or augmenting the dataset with synthetic images or additional raw data sequences.

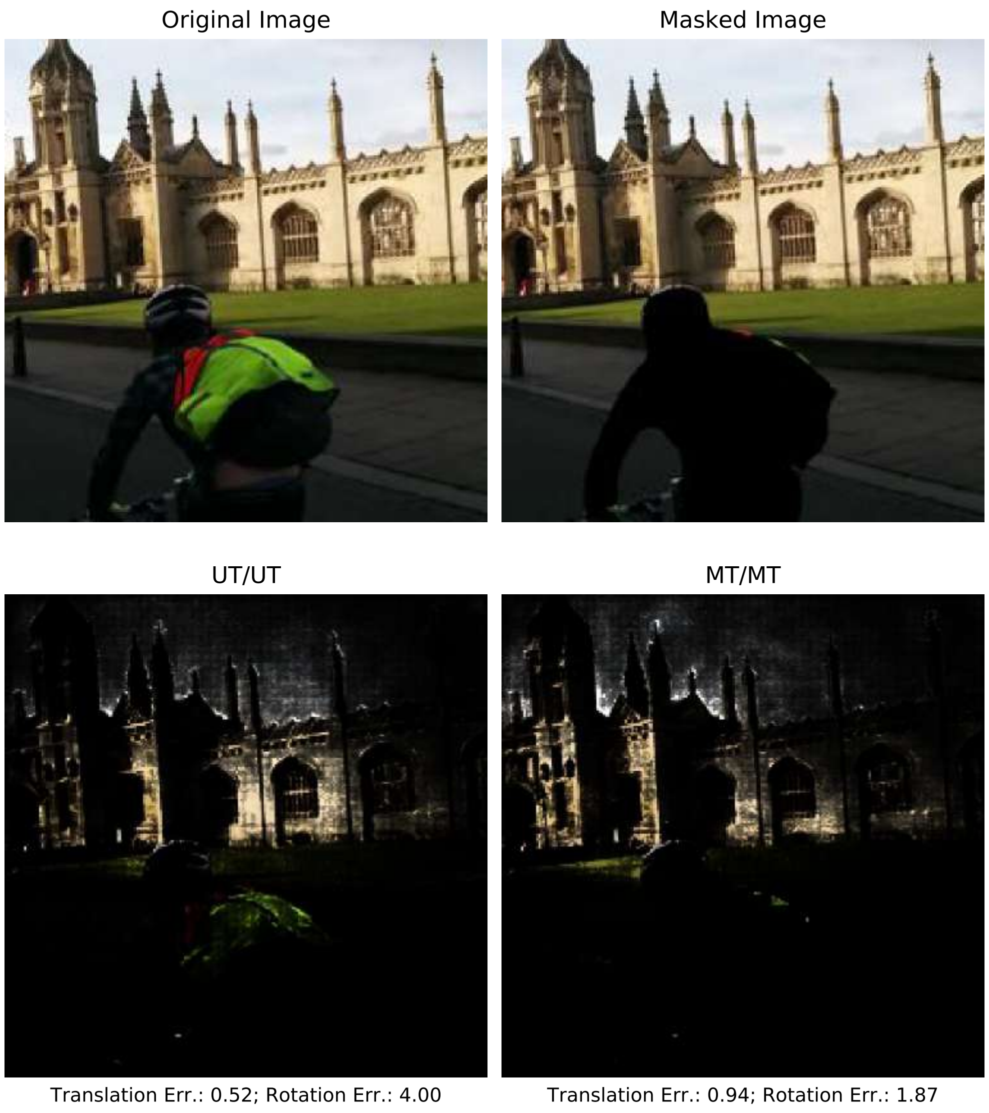

Original Image
Masked Image
UT/UT
MT/MT
Translation Err.: 0.52; Rotation Err.: 4.00
Translation Err.: 0.94; Rotation Err.: 1.87

Figure 4.12: Saliency maps with a person in the foreground

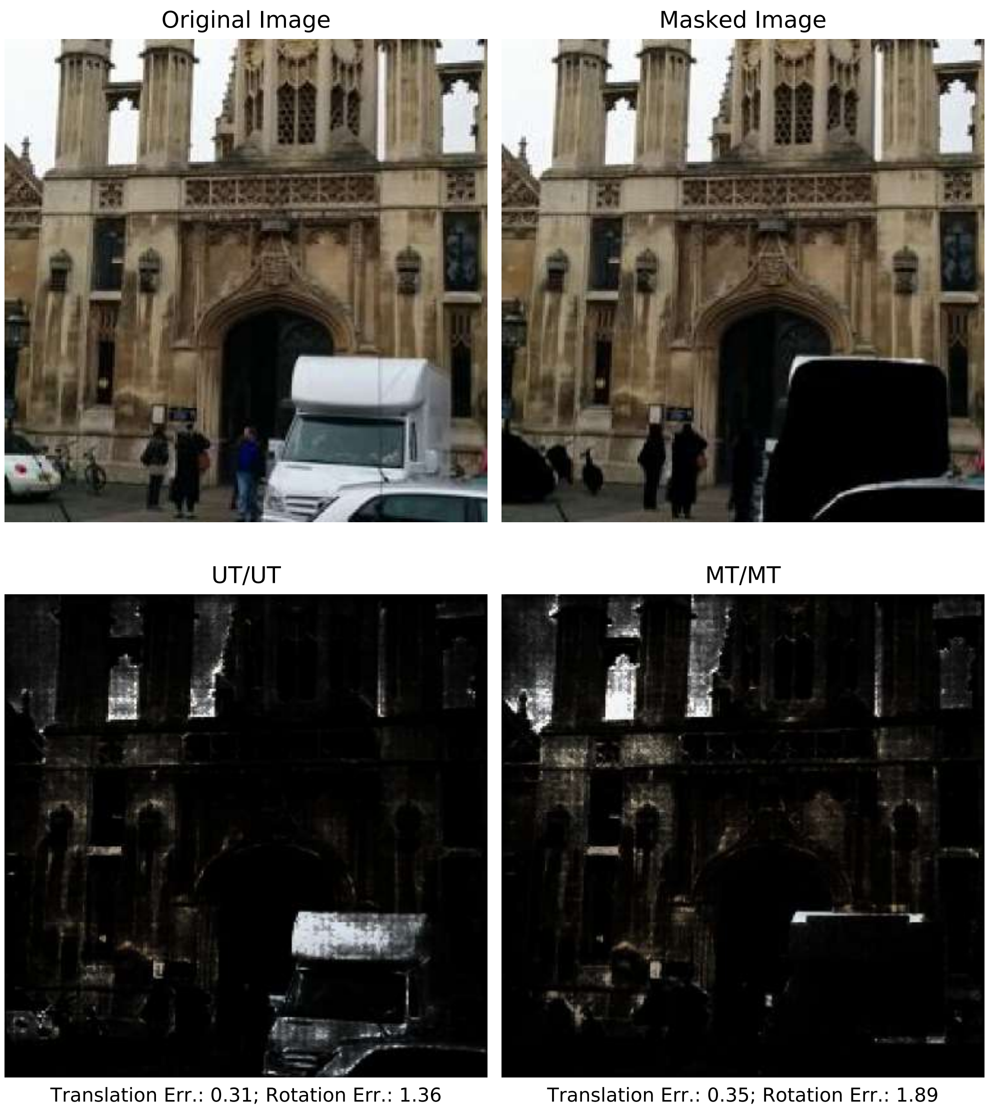

Original Image
Masked Image
UT/UT
MT/MT
Translation Err.: 0.31; Rotation Err.: 1.36
Translation Err.: 0.35; Rotation Err.: 1.89

Figure 4.13: Saliency maps with vehicles in the background

4.5 Summary

In this chapter, we applied an efficient CNN to solve the global localisation problem. In particular, we combined MobileNetV2 architecture with an MLP to estimate the 6 DoF pose. Moreover, we compared multiple design options for the MLP by symmetrically splitting the neurons into fully connected layers for learning the orientation and rotation separately. Comparison with other SotA methods using a single CNN for direct pose regression shows that our method achieves competitive results despite using a shallower network for feature extraction. Moreover, from an ablation study performed on one indoor and one outdoor environment, we noted that a monolithic MLP design, *i.e.*, sharing the weights between translation and rotation regression layers, outperforms the others in smaller scenarios. Instead, separating the last regression layers for outdoor environments obtained slightly better results. Lastly, using an NVIDIA Jetson TX2 board, we showed that the proposed approach runs faster than other methods without sensibly reducing the localisation accuracy. Subsequently, we addressed the problem of the global camera localisation in the case of dynamic object presence when using a CNN for pose regression. To this end, we proposed pre-processing images with an object segmentation network. Hence, the pre-processed and original datasets have been used to train two localisation CNN models. Through the statistical analysis, we showed that the performances of the two training approaches are similar, with a slight reduction of the error when hiding occluding objects from the views. Ultimately, we used the SmoothGrad technique with Integrated Gradients to create saliency maps highlighting the pixels contributing to the pose loss. Therefore, inspecting the obtained figures, we argue that, whilst the pose estimation would benefit overall from masking the pedestrians and other dynamic objects, the CNNs can inherently extract salient features through learning.

5

Intrusion Detection in Outdoor Environments

In this chapter*, we tackle the problem of intrusion detection in a restricted access zone around a building or, more generally, in a confined urban area. As mentioned in the introduction (see section 1.4 for a brief review of object detection), we frame the problem of aerial surveillance applications as detecting people from colour images and locating them inside an outdoor navigation environment. Whereas approaches for solving the relative and global localisation of an UAV have been proposed in the earlier parts of the present thesis, herein, we discuss the last perception elements for fulfilling the outlined autonomous aerial surveillance system's requirements. Also, similarly to the localisation tasks, the drone sensor system is limited by a single RGB camera. Such constraint is a factor that hampers the capability of 3D geometry understanding and causes the reconstruction algorithms to require further information to establish actual people's locations on a metric scale. The proposed methodology proceeds in two steps to obtain this information in more detail. First, an object detection algorithm predicts the presence of people. Specifically, it encloses the image area that tightly contains the pixels of the detected person shape with 2D bounding boxes. This procedure is explained in more detail in section 5.2. Consequently, projecting the 2D bounding box on a given ground plane, we find the 3D location of the person in section 5.3. Finally, in section 5.4, we use SfM to create a 3D reconstruction of the environment, which is fundamental for retrieving the parameters of the ground plane and for training the robot self-localisation algorithms with ground truth pose labels. Then, we could inject metric information to scale the map and obtain people's distances at this stage. Notably, the metric scale is of paramount importance when the complete system envisages an autonomous flight control based on metric units and needs to give an interpretable patrol report to the human operator. Otherwise, there would be no concern about the scale to exclusively check violations of a restricted access area. Finally, the proposed methodology results are presented on two reconstructed outdoor environments, one of which has been collected as part of this project to emphasise the localisation capabilities of the previously introduced approaches. Hence, we conclude by visualising odometry and localisation results on the 3D map with projected human body shapes.

*The work presented here has not been published yet but will be submitted for review to a journal. However, related articles have been published in collaboration with Cazzato *et al.*. in the *Sensors* journal [33], in the proceedings of the *15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, and in the proceeding of the *International Joint Conference on Computer Vision, Imaging and Computer Graphics* [34].

5.1 Related Work

In the recent literature, few works directly address the perception problems in the context of surveillance assisted by an aerial robotic platform with a monocular camera. For example, Perera *et al.*. [214] use the R-CNN method for detecting humans and fine-tuning AlexNet for processing salient image regions proposed by the EdgeBoxes algorithm [328]. Then, they train 64 4-class SVM to classify the human body rotation from HOG features extracted on the detected bounding boxes. Notwithstanding, they do not attempt to obtain the distance or even the full translation vector of the human pose with respect to the drone's point of view. Thus, they fly at known heights and with fixed camera tilt angles to experiment on human trajectory tracking. Similar to our approach, Zhang *et al.*. [315] use SVO monocular visual odometry [84] to track drone motion and estimate sparse depth maps. Hence, they obtain denser maps using PatchMatch Stereo [18], which are necessary to measure multiple averaged depth values at the bottom of detected people's bounding boxes that are assumed to lie on the ground. Hence, they find the ground plane normal vector $\mathbf{n}^\top = [n_1, n_2, n_3]^\top$ by solving the linear system with Cramer's rule and project again the bounding box's bottom centre point $\mathbf{b} = [u, v, 1]$ on the plane to find the person 3D location $\mathbf{c}$ with the following equation:

$$c = \pi_G^{-1}(\mathbf{b}) = \frac{h_{cam}\mathbf{K}^{-1}\mathbf{b}}{\mathbf{n}^\top \mathbf{K}^{-1}\mathbf{b}}\ , \tag{5.1}$$

where $\mathbf{K}$ is the intrinsic camera matrix, and h_{cam} is the camera height defined by one point $[x, y, z]$ on the ground and pitch angle θ as $h_{cam} = y\cos\theta - z\sin\theta$.

Compared to this work, the present methodology leverages SfM to reconstruct the navigation environment accurately. Currently, we do not employ depth maps like those learned by RAUM-VO (see chapter 3), but it provides a more reliable drone position to locate the human in 3D. Also, we do not require multiple detected people to estimate the ground plane robustly with the proposed approach. Further, we note that Equation 5.1 is probably erroneous, and we correctly reformulate the problem equations for projecting the point on the plane without needing the calculation of h_{cam} in section 5.3.

5.2 2D Object Detection

The object detection task involves enclosing image regions that contain an object with bounding boxes of various sizes that fit as tight as possible to the object's shape. Then, once positive boxes are found, a classifier marks them with the object's type label. Since we are interested only in people detection, the person is the only positive class, while other objects that may be present and the background are considered negative examples. To this aim, an object detection network has been implemented considering the recent SotA approaches to improve the accuracy of the predictions. So, we describe the full methodology in this section.

For the same reason for localisation and many other computer vision tasks, DL dominates the SotA literature on object detection. A Convolutional Neural Network processes images and produces high dimensional feature representations at every stage, or network's layer, of computation by gradually reducing the width and height dimensions to increase the channels. Therefore, in the early layers, small and contiguous regions of the image are processed to produce shallow, intermediate representations, but that retain the scale of the input with more fidelity, whereas, in the deeper stages, more abstract and complex features are formed at the cost of lower resolutions maps.

In early DL approaches, the classifier and the bounding box's coordinates regressor are usually connected to the last network's layer, thus having a reduced resolution that would not ease the capture of smaller objects. However, as in the SSD approach [177], including initial layers contributions may be marginally beneficial as the additional features contain less information than more deep levels, even if the input resolution is better preserved. For this reason, FPN [172] has been introduced to give the same rich semantic content to different levels of the network. To achieve their intent, the authors proposed to add a parallel network structure that would process the features of the last k levels again from top to bottom, connected with lateral pathways to the main CNN backbone. The top-down structure, illustrated in Figure 5.1, combines features from the standard feed-forward bottom-up pathway with new feature maps generated by an up-sampling operation to match the increasing resolution.

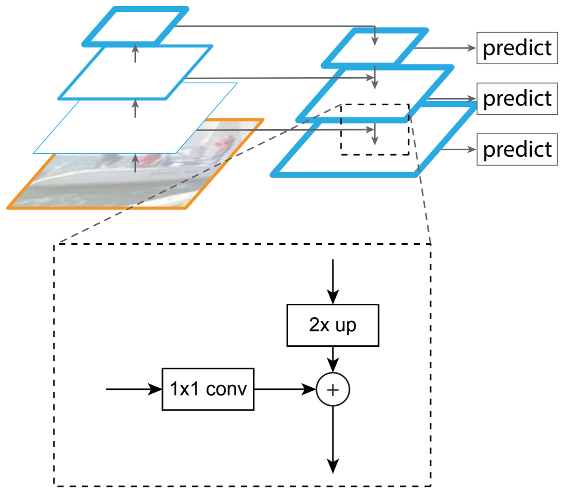


Figure 5.1: Feature Pyramid Network illustration

The figure shows the top-down feature pyramid. The highlighted block zooms over the 2x up-sampling and the 1 *x* 1 convolution to match the channel dimension of the lateral feature maps. Image from the FPN paper [172].

After introducing FPN, numerous improvements have been proposed as this design is fundamental to obtaining multi-scale detection precision. For example, PANet (Path Aggregation Network) [176] adds another bottom-up network besides the first, which behaves as a shortcut connection that shorted the path between CNN kernel recognising low-level patterns, *e.g.*, edges, and high-level semantics. Subsequent works, instead, design more complex modules to control the information flow in the pathways [170, 318] or use NAS to find the optimal configuration for top-down and bottom-up connections [94]. Taking inspiration from the previous works, Tan *et al.*. [272] introduce the BiFPN with "efficient bidirectional cross-scale connections and weighted feature fusion" as their core contribution to the SotA of object detection networks. The resulting architecture is then named EfficientDet (pictured in Figure 5.2). Their design is bidirectional, composed of repeatable top-down and bottom-up layers. Furthermore, it is cross-scale as it fuses features from multiple scales and adds extra edges to propagate the signal between same-level computation nodes. To control the information flow in the fusion process, the BiFPN learns weighting factors $w_i \geq 0$, which are then normalised either by the Softmax function or with the following more efficient "Fast normalised fusion”:

$$O = \sum_i \frac{w_i}{\sum_j w_j + \varepsilon} \cdot I_i \ , \tag{5.2}$$

where I_i is the i-th input feature, O is the output, and ε is a small constant to avoid numerical instability. Lastly, the EfficientDet design develops a family of models through a single compound scaling factor that allows to shrink or expand the BiFPN’s channels and layers jointly with the EfficientNet backbone’s width, depth, and input resolution. Hence, seven models are available in total, from $D0$ to $D6$. Due to its flexibility and multi-scale detection capabilities, EfficientDet is used in the following experiments as the base model to regress people bounding boxes.

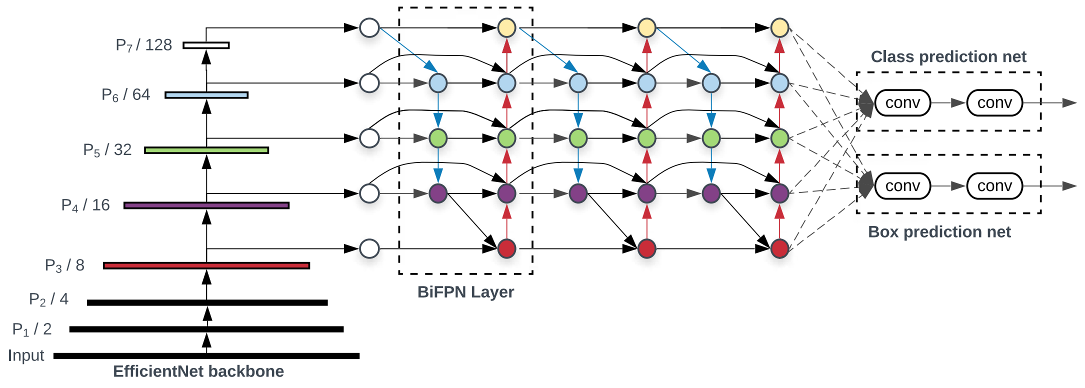


Figure 5.2: EfficientDet Architecture

The EfficientDet architecture employs EfficientNet as the backbone and a novel FPN design, namely BiFPN, to provide multi-resolution object predictions. The labels $P_i/2^i$ are associated with the i-th block output feature map of EfficientNet, which propagates into the BiFPN. Image from the EfficientDet paper [272].

A characteristic that makes EfficientDet faster is adopting a one-stage detection paradigm. As a result, it lacks a dedicated RPN as in Faster R-CNN [227]. Instead, it provides predictions for a predefined large (in the order of 10^5) set of anchor boxes that densely cover the image and present multiple possible aspect ratios and scales. This vast number of box candidates causes the effect known as the class imbalance between those spatial positions that cover objects and the majority that do not. The most common solution to this approach involves bootstrapping, *i.e.*, re-sampling data from a low represented class, or hard negative mining, which samples fewer hard examples of the most represented class. However, Lin *et al.*. [173] propose to down-weight the easy examples while enhancing the contribution of the most difficult. Their intuition is to design a loss that behaves similarly to robust losses, such as the Huber function, in reducing the effect of outliers but addressing the opposite use case. Hence, they extend the Cross-Entropy loss used to classify bounding box regions into object types by most object detection works with a novel Focal loss. Due to its remarkable effect in facilitating training and obtaining better results, we implement this loss for the object classification task.

Before introducing the Focal loss equation, we must clarify how the standard approach categorises bounding boxes. The first step is to map the logits quantities produced by the network to the positive class probability p, *i.e.*, that of being a person, by normalising the values with the Sigmoid function (see Equation 2.4). We deal with a binary classification task between one foreground object type and the background in practice. Therefore, we define the Binary Cross Entropy (BCE) loss between the network prediction p and the ground truth label y as follows, borrowing the convention used in the Focal loss paper:

$$\text{BCE}(p, y) = \text{BCE}(p_t) = -log(p_t) \ , \tag{5.3}$$

where p_t is:

$$p_t = \begin{cases} p & \text{if positive class or } y = 1, \\ 1 - p & \text{otherwise } . \end{cases} \tag{5.4}$$

An easy way to balance the weight of the positive class with the vast majority of negatives, *e.g.*, background regions, is to add a factor $\alpha \in [0, 1]$ that is inversely proportional to the frequency of the class.

$$\text{BCE}(p_t) = -\alpha \cdot log(p_t) \ . \tag{5.5}$$

However, the Focal loss intends to address the imbalance between easy and hard examples, thus focusing more on the challenging data that results in higher loss values. Therefore, a tunable hyper-parameter $\gamma \geq 0$ is added to down-weight the importance of easy examples:

$$\text{Focal}(p_t) = -(1-p_t)^{\gamma} \cdot log(p_t) \ . \tag{5.6}$$

Eventually, we can combine the two modulating factors in a single loss to achieve a more substantial counter effect over class imbalance causes:

$$\text{Focal}(p_t) = -(1-p_t)^{\gamma} \cdot log(p_t) \ . \tag{5.7}$$

The hyper-parameter α is set to 0.8 for the positive class and γ to 1.2 unless stated otherwise.

Adjacent to the convolutional structure that predicts the class logits, EfficientDet architecture includes a similar network that regresses an offset between an anchor box and the ground truth object box. The anchors compose a set $\mathcal{A}$ of default boxes scattered over the entire image plane. Hence, the last layer of the regression network produces a feature map with $W \times H$ (width and height) spatial dimensions and $4 \times A$ channels, where A is the number of anchors per spatial position and 4 is due to the $[x, y]$ coordinates of the anchors' corners offset. In practice, we use three aspect ratios $\{1{:}1, 1{:}2, 2{:}1\}$ and three scales $\{2^0, 2^{1/3}, 2^{2/3}\}$ for each of the five pyramid levels. Then, given a set of ground truth boxes $\mathcal{Y}$, each anchor $a \in \mathcal{A}$ is either unassigned or assigned to a single box $y \in \mathcal{Y}$ based on the IoU overlap metric, also known as the Jaccard index:

$$\text{IoU}(a, y) = \frac{a_{area} \bigcap y_{area}}{a_{area} \bigcup y_{area}} \ . \tag{5.8}$$

With a slight abuse of notation, we denoted with a_{area} and y_{area} the area of the anchor boxes and that of the ground truth label, respectively.

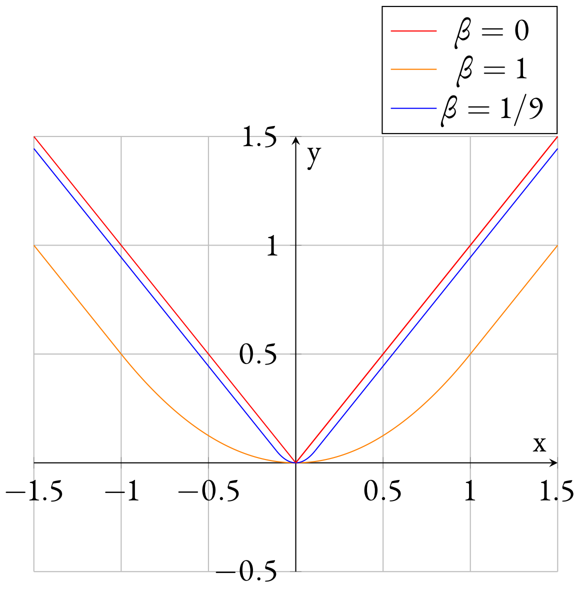


Figure 5.3: SmoothL_1 loss plot

The SmoothL_1 loss used for bounding box coordinates regression extends the Huber loss and is controlled by the hyper-parameter β. The latter differs by a factor ∂ multiplied on both cases of Equation 5.9. When the hyper-parameter β converges to zero, the loss corresponds to the simple L_1 loss. Instead, it is equivalent to the Huber loss for $\beta = 1$. For large β values, the SmoothL_1 tends to constant 0, in contrast with the Huber that gets closer to the Mean Squared Error (MSE) loss.

An anchor is assigned to the label with the maximum overlap if IoU > 0.5. Otherwise, it remains unmatched and contributes to the set of negatives or background areas discarded for regression but used for classification, as explained before. Then, we can compute the regression loss using the following SmoothL_1 function:

$$\text{SmoothL}_1(a_{coords}, y_{coords}) = \begin{cases} \frac{0.5 \cdot |a_{coords} - y_{coords}|^2}{\beta} & \text{if } |a_{coords} - y_{coords}| \leq \beta \ , \\ |a_{coords} - y_{coords}| - \frac{0.5}{\beta} & \text{otherwise} \ , \end{cases} \tag{5.9}$$

where a_{coords}, resp. y_{coords}, denotes the anchor box, resp. target box, corners' coordinates. The β hyper-parameter controls the shape of the SmoothL_1 as shown in Figure 5.3, and we set it to $1/9$.

We discard all the boxes with a confidence threshold during online inference, *i.e.*, the probability p, lower than 0.05. Consequently, the Non-Maximum Suppression (NMS) is applied to post-process the predicted boxes that are usually more numerous than the actual quantity of objects present in the scene. NMS is a technique to remove redundant boxes whose overlap area with higher confidence boxes is over a certain IoU threshold, usually 0.5.

Having established our methodology for detecting people on the 2D image plane, we move on to describe the process for obtaining their 3D location in the next section.

5.3 3D Object Localisation

In Figure 5.4, the relationship between the drone camera transformation and the object position is represented in an abstraction of the 3D world. The main assumption to obtain a 3D person location in this environment, besides an object detection algorithm, are that we know the 6 DoF pose transformation from the drone camera frame into the world frame $\mathbf{T}_c^w$. Also, we need the ground plane parameters in the 3D space, which are its normal $\mathbf{n}^\top = [n_1, n_2, n_3]^\top$ and the distance d from the origin of the coordinate system.

Then, for every point $\mathbf{p} = [x, y, z]$ lying on the plane, the following equation holds:

$$\text{Plane} : \mathbf{n}^\top \mathbf{p} + d = n_1 \cdot x + n_2 \cdot y + n_3 \cdot z + d = 0 \; . \tag{5.10}$$

To obtain the plane parameters $\mathbf{n}^\top$ and d, we rely on a sparse 3D model of the environment (see section 5.4), where we can set the ground level manually. The ground level does not always correspond to $z = 0$, especially if the axis origin is aligned with the earth's altitude. Nevertheless, the ground level allows filtering the points cloud with the relevant portion, which is selected within ± 1 meters of distance from the ground level in the z axis direction. So, using RANSAC robust fitting scheme, we can find the plane parameters that best explain the selected points with a maximum error distance of 0.4 meters for considering them inliers. Notably, it would be possible to retrieve a ground plane from dense depth maps estimated by a neural network with the same procedure since we are assuming to know the camera's pitch and height. However, this would be possible only when a portion of the ground is visible. Also, to ensure the stability of the estimation, we would require a more complex and robust

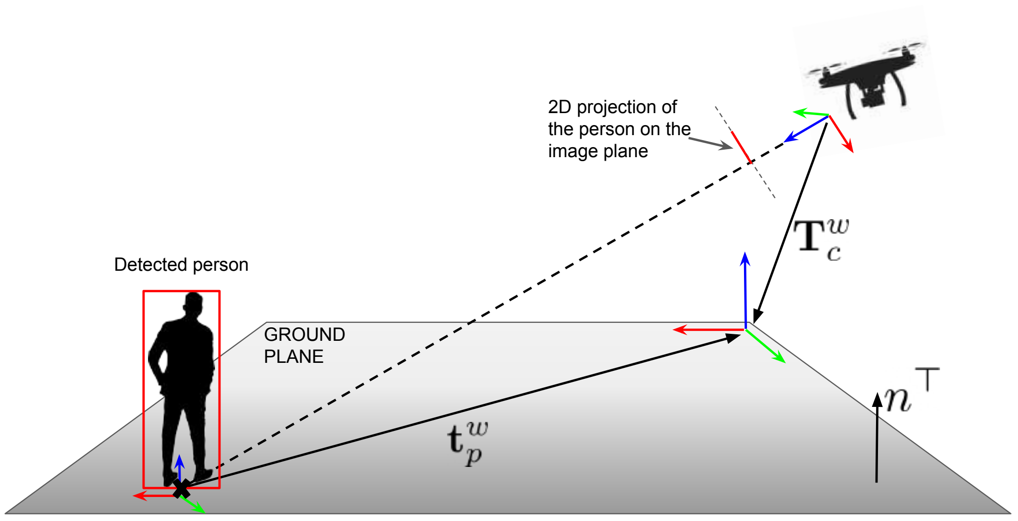


Figure 5.4: Representation of the 3D object localisation

A person is first detected on the 2D image plane using a CNN for object detection to regress the bounding box coordinates. Then, the intersection between the ground plane and line starting from the camera origin and passing by the bounding box's middle point determines the 3D location of the person. Hence, knowing the transformation $\mathbf{T}_c^w$ from the drone's camera and the world coordinates systems, we can derive the translation vector $\mathbf{t}_p^w$ that represents the person's position in the environment. Notably, this method does not estimate human body orientation.
The represent coordinate systems axis uses the $xyz \mapsto$ RGB colour mapping convention.

model, *e.g.*, tracking the ground measurements within a factor graph.

Since the plane is already defined in the world coordinates, we need to transform the bounding box base central point $\mathbf{b}_c = [u, v, 1]$ from the image to the camera coordinates and convert it into the world frame. Notably, the point is expressed in 2D homogeneous coordinates. Also, it represents the 3D vector that origins from the camera and go to infinity, or, in other words, the line that intersects the ground plane passing by the person's base location. First, however, we need to obtain the line in the same world reference of the plane to find the sought intersection point. To this aim, we use an operation $\mathcal{H}(\cdot)$ to lift the vector to 3D homogeneous coordinates to apply the intrinsic matrix, denoted with $\mathbf{K} \in \mathbb{R}^{3\times3}$, and the extrinsic matrix $\mathbf{T}_c^w \in \mathbb{R}^{4\times4}$ correctly:

$$\mathcal{H} \colon \mathbb{R}^3 \to \mathbb{R}^4;\ [x, y, 1] \mapsto [x, y, 1, 1]\ . \tag{5.11}$$

So, the point $\mathbf{b}_c$ can be transposed and rotated from camera to world coordinates applying the camera to world transformation matrix $\mathbf{T}_c^w$ after transforming pixels coordinates to camera coordinates using the intrinsic matrix $\mathbf{K}$:

$$\mathbf{b}_w = \mathbf{T}_c^w \mathcal{H}(\mathbf{K}^{-1}\mathbf{b}_c)\ . \tag{5.12}$$

Moreover, the difference between the point $\mathbf{b}_w$ and the camera translation vector $\mathbf{t}_c^w$ from the map origin returns the line direction in world coordinates:

$$\mathbf{v}_w = \mathbf{b}_w - \mathbf{t}_c^w\ . \tag{5.13}$$

Thus, knowing that a point on the line can be obtained by specifying a parameter $t \in \mathbb{R}$ in the line equation:

$$\text{Line} : \mathbf{p} = \mathbf{b}_w + t \cdot \mathbf{v}_w\ , \tag{5.14}$$

we compute the t value for which the line intersects the plane, by substituting $\mathbf{p}$ in Equation 5.10 with the right-hand side of Equation 5.14:

$$\mathbf{n}^\top(\mathbf{b}_w + t \cdot \mathbf{v}_w) + d = 0 \ , \tag{5.15}$$

$$t = -\frac{\mathbf{n}^\top \mathbf{b}_w + d}{\mathbf{n}^\top \mathbf{v}_w} \ , \tag{5.16}$$

Finally, we can find the translation vector $\mathbf{t}_p^w$ representing the location of the detected person:

$$\mathbf{t}_p^w = \mathbf{b}_w - \frac{\mathbf{n}^\top \mathbf{b}_w + d}{\mathbf{n}^\top \mathbf{v}_w} \cdot \mathbf{v}_w \tag{5.17}$$

Therefore, the described approach permits obtaining an approximation of the detected person's location in 3D, whose precision is subjected to the estimated ground plane accuracy and limited by the assumption that the base of the bounding box lies on the ground close to the person's feet. Enhancing this method aspect would be possible with an algorithm for estimating the human body's 2D keypoints, *e.g.*, OpenPose [28], as proposed in Cazzato *et al.*. [34, 35] for detecting UAV pilot using the estimated arm joint angles. In practice, OpenPose would predict the position of the person's feet in the image and would also allow predicting the body orientation after the appropriate 3D projection. However, the OpenPose's network would introduce a significant amount of additional computations, and its benefits have not been weighted in the perspective of this work's objective.

5.4 Environment 3D Reconstruction

In section 1.2, we described the SLAM problem as a MAP estimation. Besides, the MAP inference derives from a mathematical formulation based on the graphical model called the "factor graph". Notably, it encodes the optimisation variables in the nodes, *i.e.*, the robot's states and map keypoints. Instead, edges form an adjacency set of nodes, where the sensors' measurements are parameters of a function of the adjacent states, the factor. The factors are the basis for factorising a complex global optimisation function into a product of independent elements. As a result, the factor graph model visualises the problem's computational structure and the relationships between all the optimisation variables. For example, in SLAM, the graph resulting from any trajectory track reveals the sequential nature of the problem, with odometry measurements linking pairs of keyframes, landmark points appearing along the path, and revisited locations creating loop closure connections.

Unlike SLAM, reconstructing a 3D environment usually involves an unordered collection of images, sometimes captured at various times, and possibly even using different cameras. Structure from motion (SfM) is the most popular strategy for modelling the motion and the geometrical structure from unrelated images. Thus, the lack of a chronological sequence and real-time computation requirements are the principal distinctions between SLAM and SfM, extending its capabilities to map large outdoor environments from heterogeneous sets of pictures [4]. As a result, the mathematical formulation of the SfM problem is closely related to SLAM, but with a few different constraints also noticeable through its factor graph model.

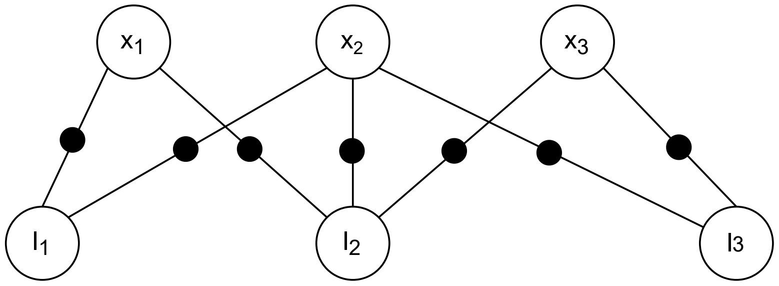


Figure 5.5: Factor graph representing the SfM problem

This figure illustrates a typical but simplified case of SfM through a factor graph model, similar to that described in Dellaert's review article on factor graphs [59]. In this example, there are three pose variables $\{x_1, x_2, x_3\}$, and three landmarks $\{l_1, l_2, l_3\}$. No odometry measurements are available to connect the poses. Still, only landmarks' observations constrain them to the environment.

The computational structure of a typical SfM problem, a simplified version shown in Figure 5.5, contains connections between camera poses to mapped landmarks 3D points, which can be clustered based on their co-visibility. These rise from visual observations of the 3D scenes in the overlapping parts of frames' FoV and are fundamental to constraining the problem of understanding the camera pose and the map structure. However, we lack the notion of odometry measurement, *i.e.*, the camera's change of motion between keyframes, which would provide another source for constraining the problem. Hence, landmarks' observations need to be collected by performing visual feature extraction and have to be matched to find the co-visibility relationships. Those images without or with few correspondences would be disconnected from the graph or would not have enough measurements to obtain a satisfactory pose optimisation result.

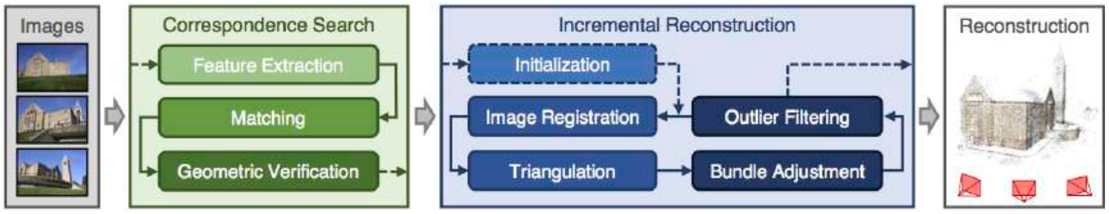


Figure 5.6: COLMAP sparse reconstruction pipeline

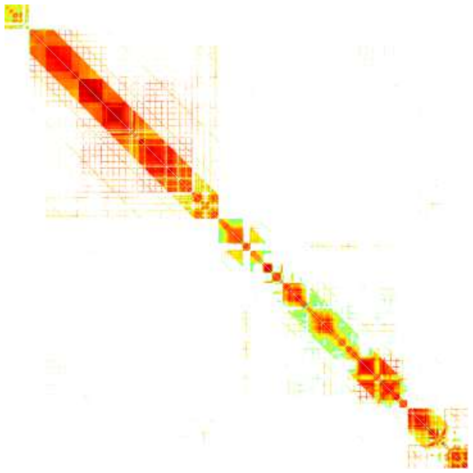
The figure shows the steps to create a sparse reconstruction of a large outdoor environment from a set of images. The first block of steps involves the extraction of 2d features from each image that are matched to find the image views that share common points.
Image from COLMAP website [130] and paper [247].

The complete SfM pipeline, shown in Figure 5.6 as implemented in COLMAP, undertakes, in fact, feature extraction and matching as its first steps. COLMAP* [247–249] is a popular SotA technique for incremental SfM publicly released by Schönberger *et al.*., and its implementation is employed for creating the reconstructions of our outdoor environments.

Feature extraction involves using a 2D keypoint detection and description algorithm that should be robust to various types of geometric or illumination changes. Borrowing the notation from [247], we describe the feature extraction phase as the detection of salient locations $\mathbf{x}_j \in \mathbb{R}^2$, also called keypoints, in the i-th image of the collection $\mathcal{I} = \{I_i \,|\, i = 1 \ldots N_I\}$ to be reconstructed and create a set of local 2D features $\mathcal{F}_i = \{(\mathbf{x}_j, \mathbf{f}_j) \,|\, i = 1 \ldots N_F\}$. The 2D features associate a vector $\mathbf{f}_j \in \mathbb{R}^d$ to each keypoint for describing its visual appearance with a d-dimensional code. Ideally, these codes would uniquely identify the referred spatial location to be matched with other observations of the same 3D spot. In practice, the quality of a feature descriptor is measured by its repeatability and distinctiveness properties [87]. Because we adopt the Scale-Invariant Feature Transform (SIFT) [185] method for this step, we obtain 128-dimensional vectors.

*open-source software available at https://github.com/colmap/colmap

The matching phase discovers the connections between the frames in $\mathcal{I}$ based on the extracted feature $\mathcal{F}_i$. The outcome of the step is a scene graph that joins couples of images that share an overlapping FoV based on the features in common. The brute force approach compares all the feature combinations $\mathcal{F}_a \times \mathcal{F}_b$ using a similarity metric to establish a match between two images (I_a, I_b). The quadratic complexity of this approach, *i.e.*, $\mathcal{O}(N_I^2 N_F^2)$, makes it unfeasible for extensive collections, but it ensures that all the frames are strongly connected to the graph. Nevertheless, a more efficient strategy is to rely on image retrieval techniques that speed up finding the nearest neighbours' images in the collection. We use a vocabulary tree with 256K words trained on the Flickr100k dataset [215] to perform BoVW image retrieval with spatial re-ranking [248] and visual burstiness weighting [128] (procedure implemented in COLMAP within the `vocab_tree_matcher` function). We also apply the transitive matching strategy to densify the scene graph (COLMAP `transitive_matcher` function). If we already possess an approximate knowledge of the camera position, *e.g.*, through GPS, we also apply a spatial matching with the nearest neighbours inside a certain radius (COLMAP `spatial_matcher` function). To conclude the first stage of the SfM pipeline, the matches are verified by finding a valid frame-to-frame transformation, either described by a homography or an essential matrix, that can project a sufficient amount of features from one image to the other with a low reprojection error. Figure 5.7 shows the adjacency matrices of the scene graphs resulting from the reconstruction of two environments introduced later.


(a) AU-AIR dataset

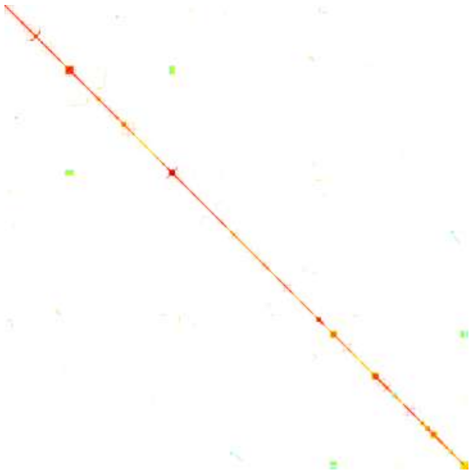

(b) Belval dataset

Figure 5.7: Plots of the adjacency matrix of the scene graph

The matrices show the matching connection between every pair of images in the two collections reconstructed in the experiments. The resulting graph does not contain any separate components. Nevertheless, it is not so dense to hamper the optimisation. The colour represents the strength of a match, where red corresponds to a higher number of features and green to a lower. The two scene graph may appear with different connection densities, but this effect is due to the more significant number of frames mapped in the Belval dataset (around 10x more) that scales the plot. The original images have a prohibitive resolution to be displayed, corresponding to a number of $N_I \times N_I$ pixels. Therefore, they have been scaled down to 2000×2000 pixels for illustration purposes.

The second stage of COLMAP is the actual incremental reconstruction of the environment. It uses the previous stage outputs, the match matrix and the set of features for registering the camera frame in an increasingly growing scene model. The process starts from a two-view motion model estimation and continues with an alternation between the triangulation and the registration steps. The first step extends the set of the map's 3D points $\mathcal{X} = \{\mathbf{X}_k \in \mathbb{R}^3 \,|\, k = 1 \ldots N_X\}$. The second step adds new camera pose estimates to set $\mathcal{P} = \{\mathbf{P}_c \in \mathbf{SE}(3) \,|\, x = 1 \ldots N_P\}$ using PnP and the correspondences between 2D features and triangulated 3D points. Finally, BA refines all pose and point measurements by minimising the reprojection error function with an iterative non-linear problem solver, such as LM. BA is a problem tightly related to the MAP optimisation of the factor graph. However, as evident, the factor types are restricted to the camera projective geometry functions, and the data is entirely available from the start of the SfM process [26]. Hence, we formulate an energy function E that sums the contributions of the distances between all 2D features $\mathbf{x}_j$ and their associated 3D points $\mathbf{X}_k$ projected on the image plane by a function π and camera parameters $\mathbf{P}_c$:

$$\mathrm{E} = \sum_j \rho\left(\|\pi(\mathbf{P}_c, \mathbf{X}_k - \mathbf{x}_j\|_2^2\right) \ , \tag{5.18}$$

where $\rho(\cdot)$ is a robust cost function to down-weight the influence of outliers, such as the Cauchy function [10]. Then, SfM alternates between local BA of highly connected parts of the scene graph and global BA to reduce the drift consequent to the model growth. COLMAP `hierarchical_mapper` function performs these reconstruction steps by partitioning the scene graph and parallelising the BA process. Then, COLMAP merges the small sub-models based on overlapping scene observations. Ultimately, we perform the last triangulation and a refinement BA with `point_triangulator` and `bundle_adjuster` functions.

At the end of the reconstruction, SfM produces a sparse model of the environments whose origin is placed arbitrarily within the scene and with a meaningless scale. Therefore, we first transform "latitude-longitude-altitude" GPS coordinates to an *xyz* coordinate frame whose origin is placed at a specified GPS position. We used the Matlab function `latlon2local` in our reconstructions, which assumes the geographic coordinates have been recorded with the standard WGS84 reference ellipsoid. Finally, using the RANSAC routine implemented in the COLMAP function `model_aligner`, we compute a 7 DoF transformation that applies a translation, rotation, and scale to align the poses estimated by SfM with those approximated by the GPS *xyz* coordinates.

5.5 Experiments and Results

In this section, we present the experiments on two outdoors environments, AU-AIR and Belval, where the latter has been created as part of this project. First, we show the 3D reconstruction of each dataset obtained using COLMAP and adapting the parameters. Next, we present the people detection and localisation using the objects portrayed in the AU-AIR dataset's images. Only a qualitative evaluation is presented since 3D object labels have not been released (only classical 2D bounding boxes are available). Then, a global localisation network is evaluated with a methodology similar to that explained in chapter 1 but adopts a different network for the feature extraction.

Furthermore, we introduce a novel dataset, Belval, created within this project's scope to highlight the capabilities of the neural network approach for the odometry estimation problem with monocular cameras compared to traditional techniques. Nevertheless, the purpose of this dataset is to recreate a scenario for the surveillance task as initially described. However, we do not yet provide 2D objects and 3D pose ground truth labels for this dataset for evaluation. Thus, the evaluation of the detection and localisation of people is left for future work.

Finally, we illustrate numerically and graphically (in the reconstructed scene) the results obtained with the odometry methodology described in chapter 3. To this aim, we use the SfM pipeline to get a reliable source of ground truth poses to benchmark the localisation method and qualitatively appreciate the estimated drone trajectories in all three dimensions.

5.5.1 AU-AIR Dataset

AU-AIR [21] dataset has been created from 8 videos captured from a Parrot Bepop 2 drone performing low-level flight trajectories (from 10 to 30 meters of height) over a large road junction (a roundabout). The dataset, meant for traffic surveillance, contains bounding boxes of eight object categories, *e.g.*, person, car, bus, van, truck, bike, motorbike, and trailer, with the majority of labels corresponding to cars, 100K, and considerably fewer people, around 5K. Notably, AU-AIR consists of $32,823$ frames shot at 30 FPS with a resolution of 1920×1080. Also, it delivers multi-modal data recordings of the drone state, such as altitude, IMU, and GPS synchronised by the given timestamp. However, this additional information may be used to test object tracking. Still, the lack of camera intrinsic parameters and more precise ground truth poses do not permit the correct localisation of the drone and the detected objects. For this reason, we provide a 3D reconstruction to recreate the missing information from the images and align the reconstructed model of the scene with the given GPS. Nevertheless, the limited pose variation in the space does not contribute to a challenging testbed for localisation algorithms.

3D Reconstruction

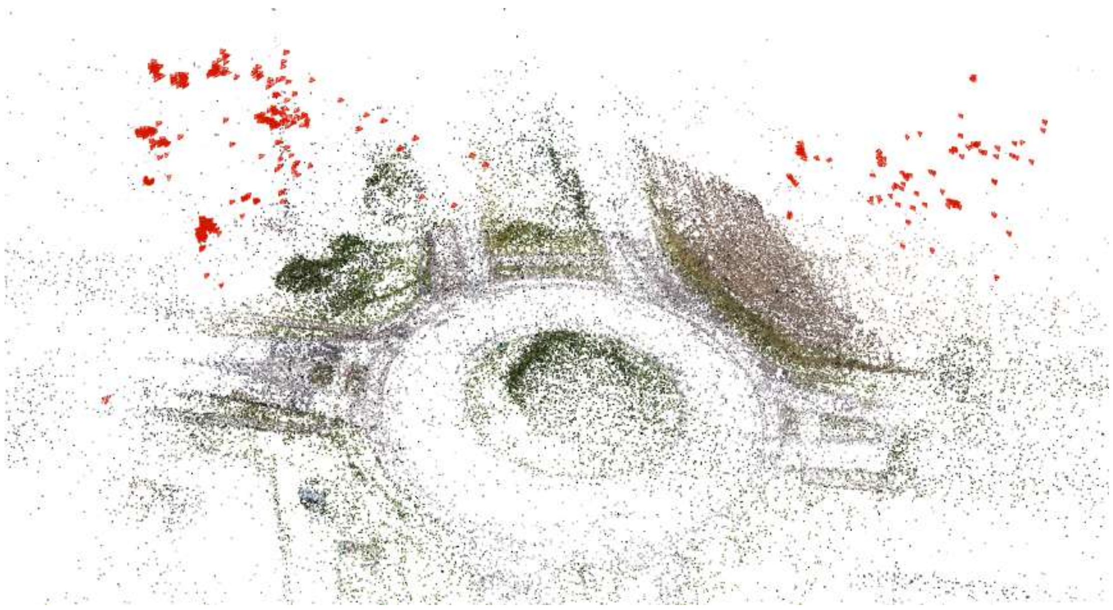
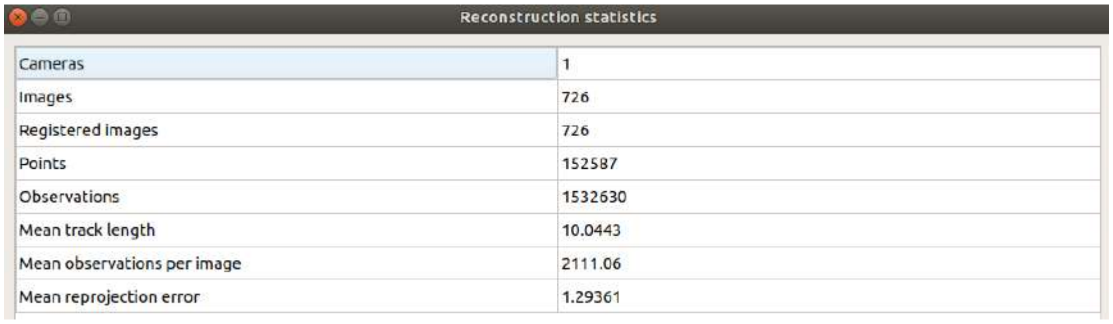


Figure 5.8: AU-AIR sparse reconstruction

3D scene model of AU-AIR obtained with COLMAP.

In Figure 5.8, we display the 3D reconstruction obtained registering only a portion of the over 32K images. In detail, after ordering all the frames chronologically, we select images based on a 1.5 seconds time interval. This process is needed to ensure enough motion between one image and another to triangulate points. In addition, the distribution of the registered camera frames highlights the issue of the poses grouped in two clusters, which results in a restricted variability of the scene observation viewpoints. Then, in Figure 5.9, we show the report of the reconstructed model.



Reconstruction statistics

Cameras	1
Images	726
Registered images	726
Points	152587
Observations	1532630
Mean track length	10.0443
Mean observations per image	2111.06
Mean reprojection error	1.29361

Figure 5.9: AU-AIR 3D reconstruction stats

Starting from the sparse reconstruction, we used COLMAP multi-view stereo [249] to create a dense point cloud model of the scene shown in Figure 5.10. Using this dense model, we fit the 3D plane parameters representing the ground plane. To compute them and display the results, we use the Open3D python library for 3D point cloud manipulation[*] [323].

[*]Open source-code available at http://www.open3d.org/

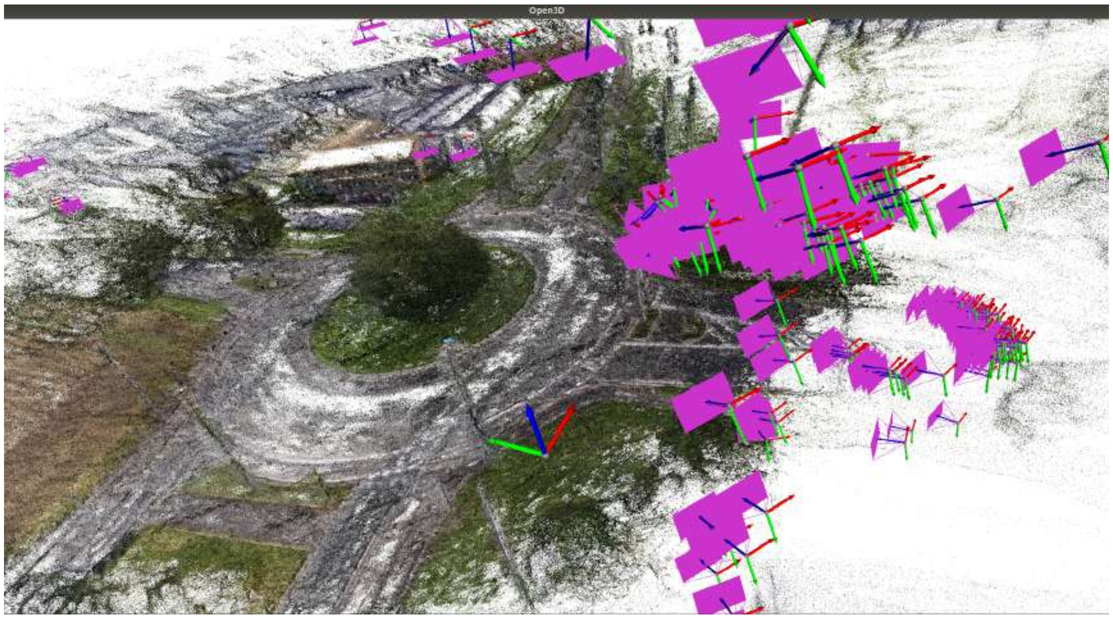


Figure 5.10: AU-AIR dense reconstruction

Dense reconstruction of AU-AIR obtained with COLMAP and multi-view stereo technique.

Object Detection and Localisation

To detect objects, we train an EfficientDet-D2 model. The suffix D2 stands for the compound coefficient 2 (chosen between 0 and 7) used to scale the model width and height properties, finding a trade-off between detection performances and efficiency. Consequently, the input images size 768×768 pixels, the BiFPN is repeated 5 times and the final object regressor and classifiers are composed of 5 convolutional layers before the final output. Also, the backbone is an EfficientNet using the same scaling coefficient. We modify an open-source implementation* and use their pre-trained weights on the MS COCO [174] dataset on all 90 object classes, adapting them to the binary classifier for our task. Among the improvements, we adapt the convolutional operation to correctly export the network's computation graph to an ONNX intermediate format. Consequently, this meta-representation is translated to the NVIDIA TensorRT model, which can run at higher speeds on embedded platforms, such as the Jetson Nano or TX2.

*https://github.com/zylo117/Yet-Another-EfficientDet-Pytorch

Furthermore, we improve the training process to enable multi-GPU deployments on distributed server clusters. We use the PyTorch [210] library as a wrapper of the NVIDIA Collective Communications Library (NCCL) to communicate between multiple NVIDIA V100 GPUs. Thanks to the University of Luxembourg's High-Performance Computing (HPC) infrastructure, we can run the network on a theoretical maximum of 18 server nodes with 4 NVIDIA V100-16 GB each plus 6 nodes with 4 NVIDIA V100-32GB each. In practice, we limit to the use 4 or 8 GPUs when needed. This training procedure permits training on large datasets, such as MS COCO with more than 60K training data samples, faster by spreading a batch of data over multiple devices. The current implementation does not allow the allocation of more than 8/16 samples in 16 GB of memory. Also, we apply the Synchronized Batch Normalization that gathers the batch statistics from all devices. then, it keeps the running average computing $\hat{x}_{new} = (1 - \text{momentum}) \times \hat{x} + \text{momentum} \times x_t$, where x_t is the tracked statistic, $\hat{x}$ its running average, and momentum is a parameter set to 0.1.

Then, we use AdamW [184] optimizer with an initial learning rate of 10^{-3} for the regressor and classifier weights and a lower 10^{-4} for the backbone. Also, we add a weight decay regularization with a L_2 penalty norm of 10^{-5}. Furthermore, we schedule the learning rate decay to follow a cosine annealing with warm restarts strategy [183]:

$$\eta_t = \eta_{min} + \frac{1}{2}(\eta_{min} - \eta_{max}) \left(1 + \cos\left(\frac{T_{cur}}{T}\pi\right)\right) \quad , \tag{5.19}$$

where $\eta_t$is the learning rate after the decay, η_{min} is the minimum learning rate set to 10^{-8} and η_{max} is the initial value. The ratio $\frac{T_{cur}}{T}$ regulates the learning rate interpolation between each restart, where T_{cur} is the current epoch number, and T is the interval of epochs before for every restart. In the experiments, we set it to 50.

Differently from the original work, we combine the cosine decay with a step function that reduces the maximum learning rate η_{max} in correspondence of every restart by a factor γ (set to 0.25). We note that restarting at the initial learning rate hampers the training because the loss jumps back to high loss values and keeps a long time to reach a minimum. We show a plot of this combined schedule in Figure 5.12a relative to the global localization experiment. Also, we multiply T by 2 after every restart to increase the interval length.

Then, we train for 15 epochs on the COCO dataset using the BCE loss and finish the training on AU-AIR for the other 10 epochs. After this period, we noticed a degradation in the evaluated test metrics. The metric used for the evaluation are the average precision (AP) and average recall (AR). As a reminder, precision and recall are defined as follows:

$$\text{precision} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalsePositive}}\,, \tag{5.20}$$

$$\text{recall} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalseNegative}}\,. \tag{5.21}$$

Then, we compute the precision at the interpolated points of the Precision-Recall curve with an IoU area threshold of 0.5% between the prediction and the ground truth. A similar procedure is used to obtain the AR, averaged in the interval $[0.5, 0.95]$ with a step size of 0.05.

Following, we show the results obtained on AU-AIR in Table 5.1:

Table 5.1: AU-AIR object detection results

	EfficientDet-D2
AP	0.401
AR	0.417

In the following Figure 5.11, we display the 3D people localisation. We cannot quantify the amount of error without the ground truth positions for this task. For the visualization, we used the ground truth position of the drone as registered by the SfM. However, it is always possible to obtain the camera pose estimate from the SfM with the registration module, even less efficiently than the proposed global localisation network.

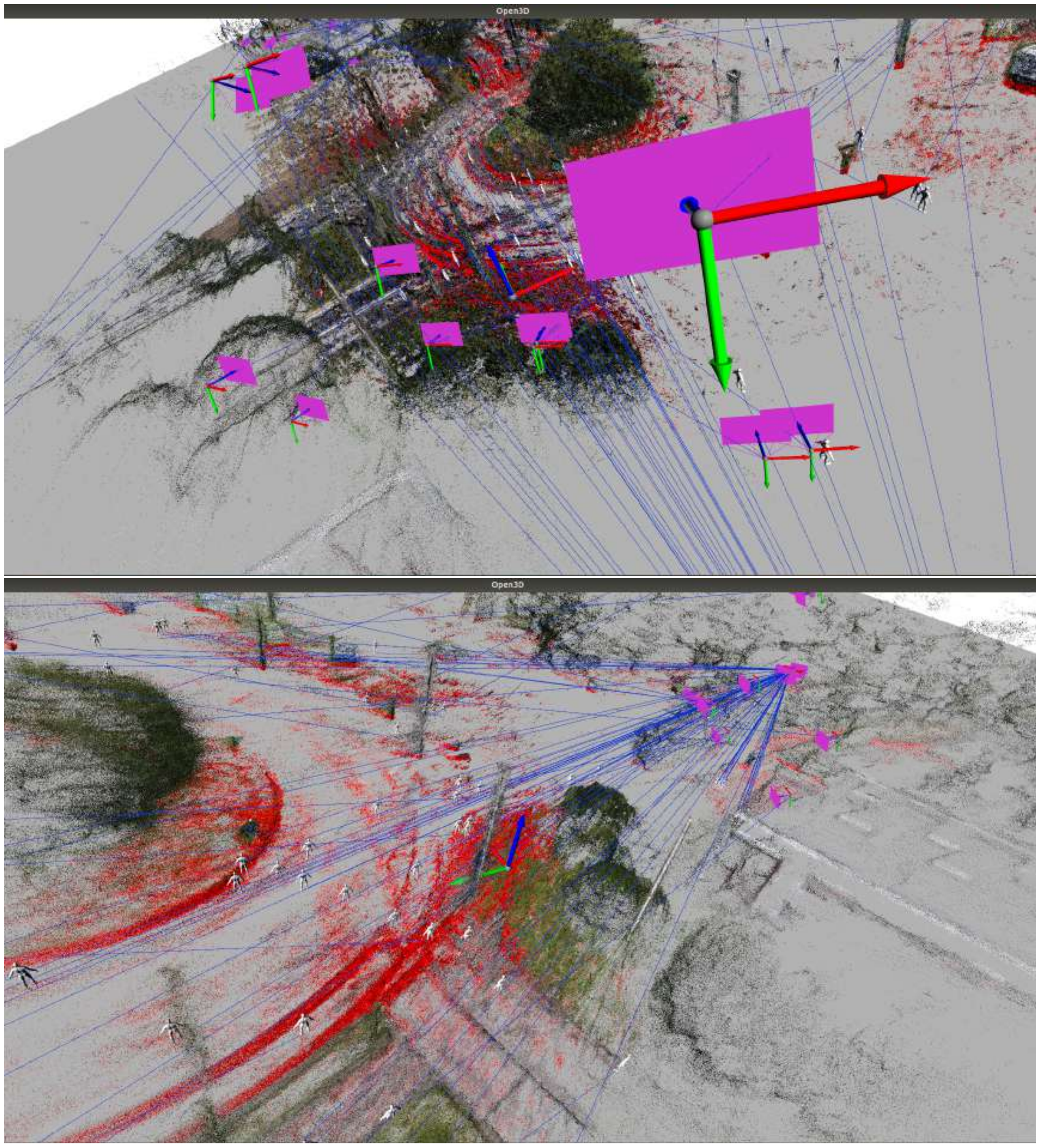


Figure 5.11: 3D locations of detected people in the dataset

Two points of view on the AU-AIR scene displaying the 3D position of the detected people detected. The grey plane is the ground. The red points are the inliers of the RANSAC parameter fitting. The blue lines, originating from the camera centre, pass from the bounding box on the image plane and end at the intersection with the ground plane, where we place a 3D mesh of the human body for illustration.

Global Localisation

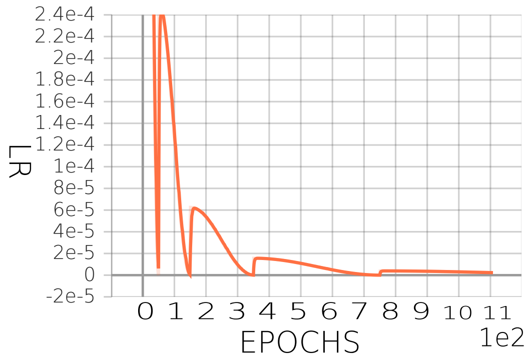
Using the camera poses registered by SfM on the AU-AIR dataset, we test the global pose localisation DL approach of chapter 4. However, we substitute the backbone network with an EfficientNet-B2, the same as for object detection. Also, the output of the network is the quaternion logarithm as proposed by Brahmbhatt *et al.*. [22]:

$$\mathbf{w} = \log(\mathbf{q}) = \begin{cases} \frac{\mathbf{b}}{\|\mathbf{v}\|} \arccos(u) & \text{if } \|\mathbf{v}\| \neq 0, \\ 0 & \text{otherwise} \end{cases}, \tag{5.22}$$

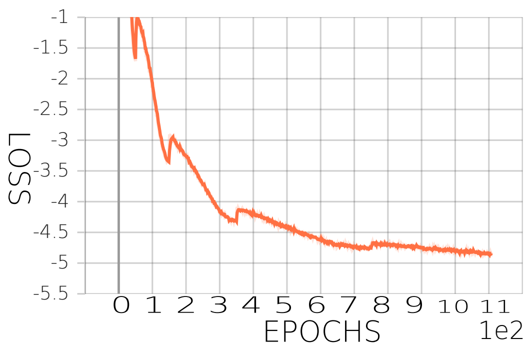
where $\mathbf{q}$ is a unit quaternion composed by a scalar part u and a vector part $\mathbf{v} \in \mathbb{R}^3$, and $\|\cdot\|$ is the Euclidean norm. The log quaternion $\mathbf{w} \in \mathbb{R}^3$ is mapped back to the corresponding quaternion with the exp function:

$$\mathbf{q} = \exp(\mathbf{w}) = \left(\cos(\|\mathbf{w}\|), \frac{\mathbf{w}}{\|\mathbf{w}\|} \sin(\|\mathbf{w}\|\right) = (u, \mathbf{v}) . \tag{5.23}$$

Then, we train using the same training distributed technique and the learning rate schedule for object detection, with an initial η set to 10^-3. We train until convergence but keep the model that achieves the lowest loss, *e.g.*, at epoch 1100 in this experiment. The loss curve in Figure 5.12b resembles that of the learning rate in Figure 5.12a, displaying a sudden increase in correspondence of the restart. We limited this effect with the integration of the step function.



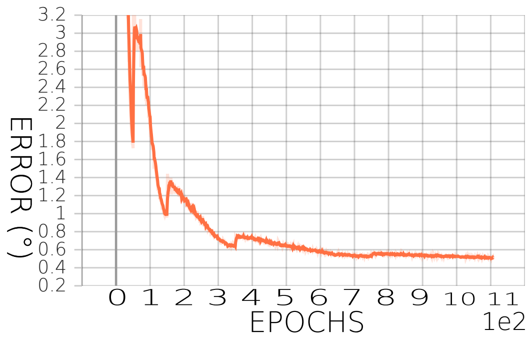
(a) Learning rate decay



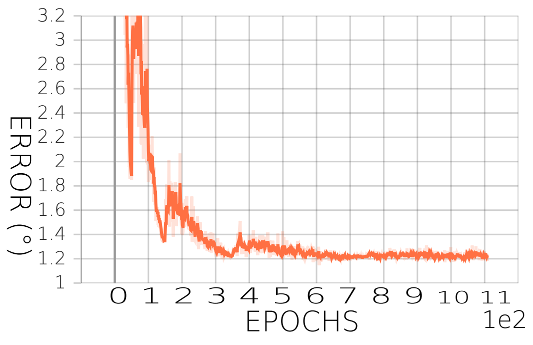
(b) Training loss

Figure 5.12: Plot of the learning rate decay and pose train loss

In Figure 5.13 and Figure 5.14, we show the median rotation and translation error during training on the train set and a small validation (val) set. The median and mean error computed on the test set are then included in Table 5.2. The proposed method allows localising the drone's position in the AU-AIR scene with an accuracy in the order of a few centimetres in most cases. Also, in Figure 5.15, the cumulative distribution of the translation and rotation errors shows that most of the frames are localised within an acceptable distance from the real pose. The orientation, which is the most difficult to predict, is also less than 5° in about the 90% of cases.

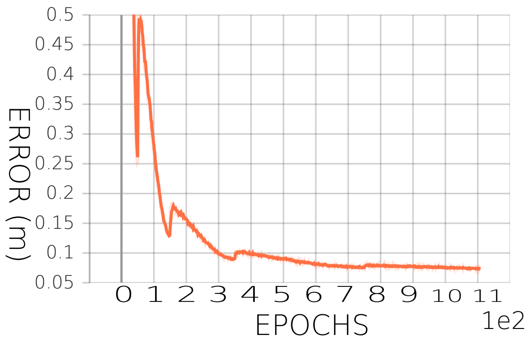


(a) Median rotation error for the train set

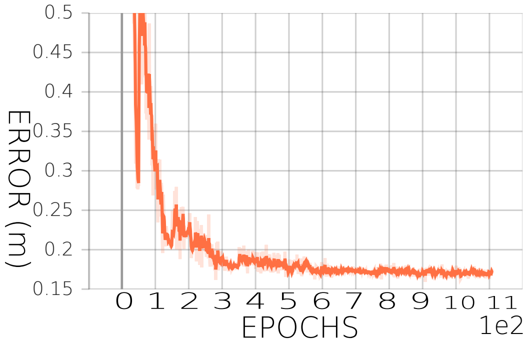


(b) Median rotation error for the validation set

Figure 5.13: Plot of the median rotation error during training



(a) Median translation error for the train set



(b) Median translation error for the val set

Figure 5.14: Plot of the median translation error during training

Table 5.2: Global Localisation Results on AU-AIR

The table includes the results of the localisation using EfficientNet-B2 for feature extraction. We compute the Median and Mean of the Translation Error and the Rotation Error.

	EfficientNet-B2
Median Translation Error	0.16 (m)
Median Rotation Error	1.22 (○)
Mean Translation Error	0.332 (m)
Mean Rotation Error	2.346 (○)

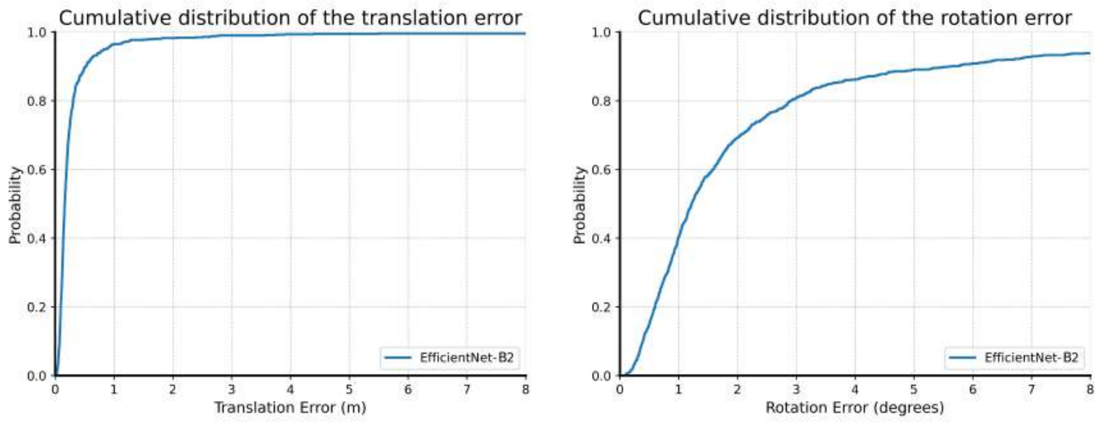


Figure 5.15: Cumulative distribution of the localisation error

5.5.2 Belval Dataset

Herein, we describe the Belval dataset[*], which takes the name from the locality in which the main building of the University of Luxembourg is located. We perform various flight sequences in a limited area confined within the university buildings on three sides. In Figure 5.16, we plot the geographic position of a manually controlled DJI Mini 2 drone, which records the meta-data with timestamps. These synchronize the drone monocular camera, which streams a video in FullHD resolution, *i.e.*, 1920 $\times$ 1080 pixels per frame, at 30 FPS. Hence, we created a set of 5 sequences, counting from 3 to 7 since the first 2 were discarded, for a total of 17955 images.

[*]Available at https://github.com/snt-arg/BelvalDataset

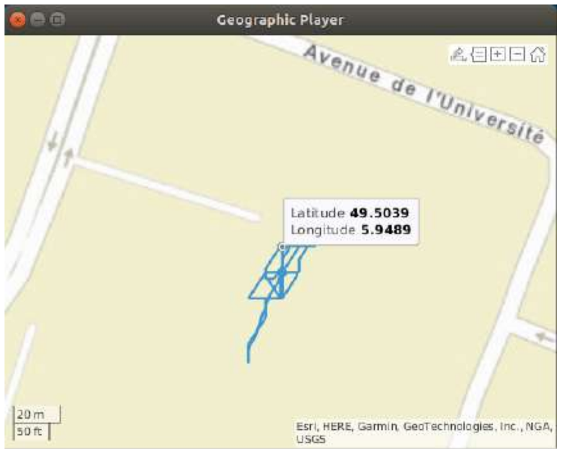


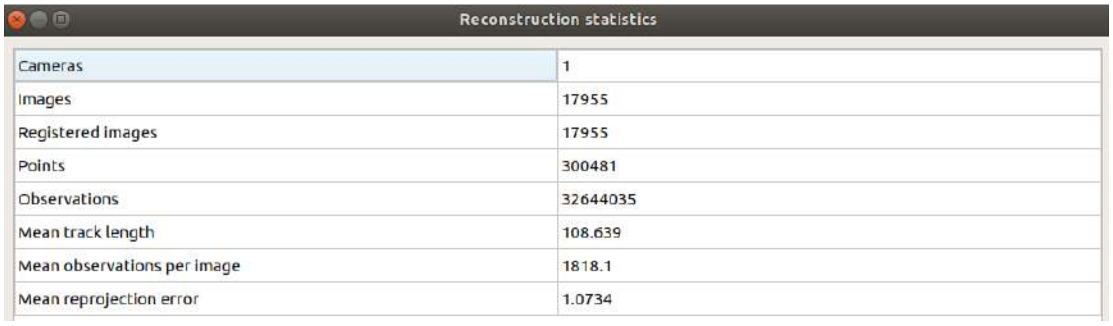
Figure 5.16: Belval dataset GPS coordinates

3D reconstruction

We have reconstructed a model of the Belval 3D scene by registering all the dataset frames. Notably, for matching improvement, we initialise the pose of the frames with the associated GPS coordinate. Also, we pre-calibrate the camera's intrinsic parameters using OpenCV. The obtained perspective transformation parameters and the non-linear radial distortion coefficients are:

$$[f_x, f_y, c_x, c_y] = [1515.911603, 1522.668902, 993.757878, 535.610019]$$
$$, [k_1, k_2, k_3, k_4, k_5] = [-0.0057670.038655 - 0.0002050.0090460.000000] \; .$$

InFigure 5.17, we show the statics of the reconstruction process, which took approximately 3 days. Figure 5.18 displays the sparse point cloud obtained at the final stage after aligning the model with the GPS. We also performed a dense reconstruction. However, due to the model's size, it was impossible to fit it into the memory for visualization.



Cameras	1
Images	17955
Registered images	17955
Points	300481
Observations	32644035
Mean track length	108.639
Mean observations per image	1818.1
Mean reprojection error	1.0734

Figure 5.17: Belval 3D reconstruction statistics

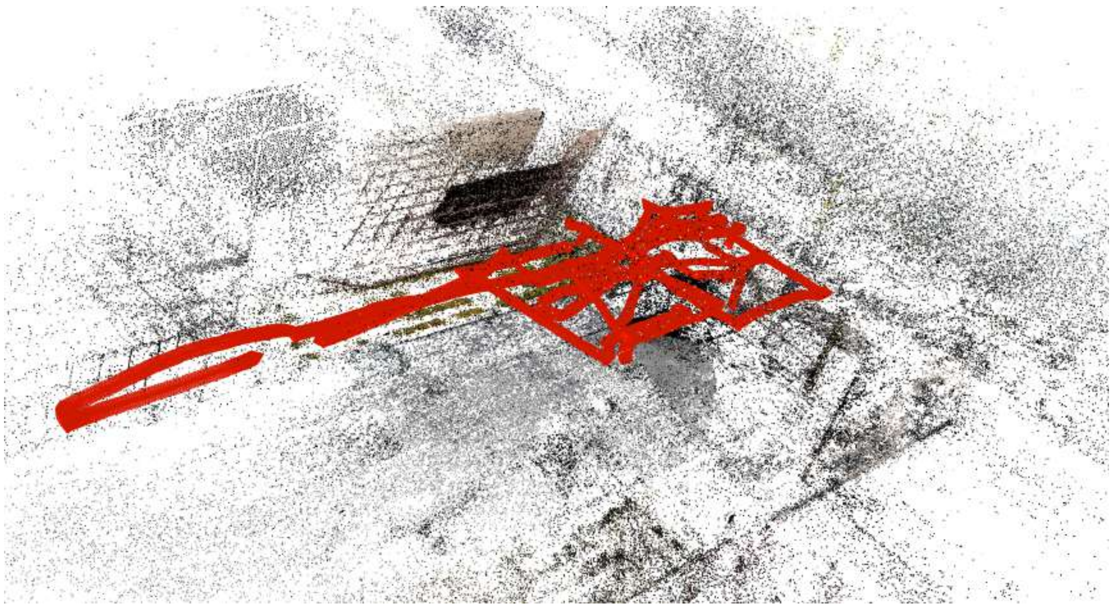


Figure 5.18: Belval 3D sparse reconstruction

Relative Localisation

In Table 5.3, we illustrate the results of the odometry estimation using the methodology described in chapter 3. We used the sequences 4 to 6 to train the models, while the other two were used only for testing. We compare the simple network training model, **Simple-Mono-VO**, the RAUM-VO model, **RAUM-VO w/ PoseNet**, which uses the translation vector estimated by the pose network, and the RAUM-VO model with translations estimated by

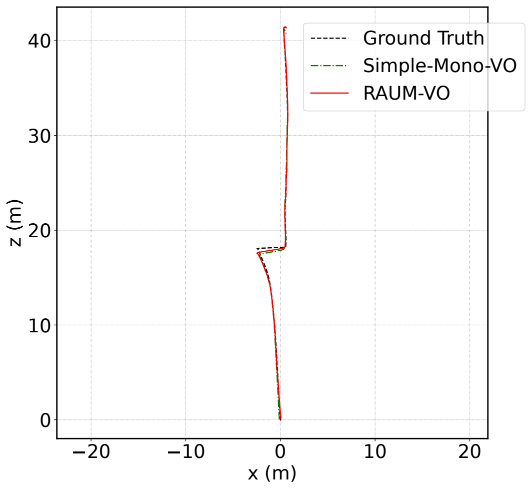
PnP, RAUM-VO model **RAUM-VO w/ PnP**. Notably, in this dataset, the poses obtained by combining the rotations from RAUM-VO and the translations from PnP outperform the pose network output. Compared to KITTI, Belval dataset sequences 5 and 6 exhibit complex trajectories. Notably, we argue that, in KITTI, VO algorithms are assisted by the holonomic movement constraint of the car and by an almost absent height variation.

Table 5.3: Odometry quantitative evaluation.

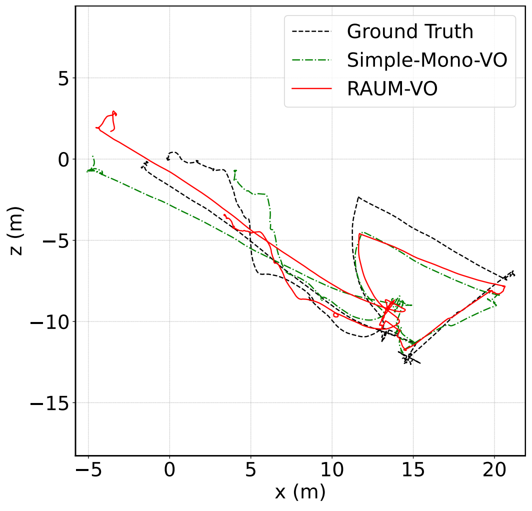
The table displays the results of the estimated ego-motion for each trajectory in the Belval dataset.

Model	Metrics	03	04	05	06	07	Avg.Train	Avg.All
Simple-Mono-VO	t_{errs}	9.380	6.697	12.031	15.762	19.638	15.810	12.702
	r_{errs}	32.777	26.745	33.257	56.701	91.224	60.394	48.141
	ATE	1.561	0.712	2.408	3.793	4.380	3.527	2.571
	RPE (m)	0.088	0.056	0.072	0.069	0.063	0.068	0.070
	RPE (°)	0.115	0.090	0.150	0.155	0.232	0.179	0.148
RAUM-VO w/ PoseNet	t_{errs}	14.843	10.771	12.708	14.731	22.347	16.595	15.080
	r_{errs}	7.870	5.451	22.885	27.116	38.768	29.590	20.418
	ATE	1.352	0.540	3.019	3.391	3.122	3.177	2.285
	RPE (m)	0.087	0.055	0.072	0.067	0.059	0.066	0.068
	RPE (°)	0.063	0.054	0.186	0.213	0.206	0.202	0.145
RAUM-VO w/ PnP	t_{errs}	4.021	4.819	11.660	14.711	9.471	11.947	8.936
	r_{errs}	7.870	5.451	22.885	27.116	38.768	29.590	20.418
	ATE	0.637	0.257	2.776	3.394	1.330	2.500	1.679
	RPE (m)	0.047	0.026	0.061	0.060	0.040	0.054	0.047
	RPE (°)	0.063	0.054	0.186	0.213	0.206	0.202	0.145

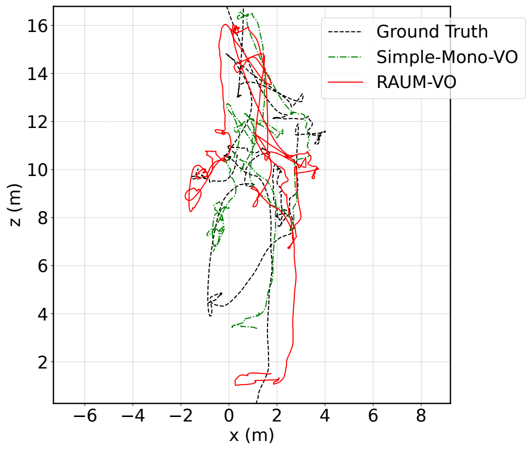
In Figure 5.19 and Figure 5.20, we plot the estimated trajectories for the train and test sequences, respectively. Note that the camera coordinate frame is rotated with respect to a classical *xy* ground plane and *z* up direction. Hence, *z* is aligned with the drone's forward direction and *x* with its right. As a result, it is possible to observe that the network models can predict the path with fewer abrupt motions. At the same time, on the most complex sequence 6, it hardly manages to keep track of the odometry. Instead, a traditional SLAM method, such as ORB-SLAM, could partly estimate the odometry only in the simplest sequences 3 and 4.



(a) Seq. 04

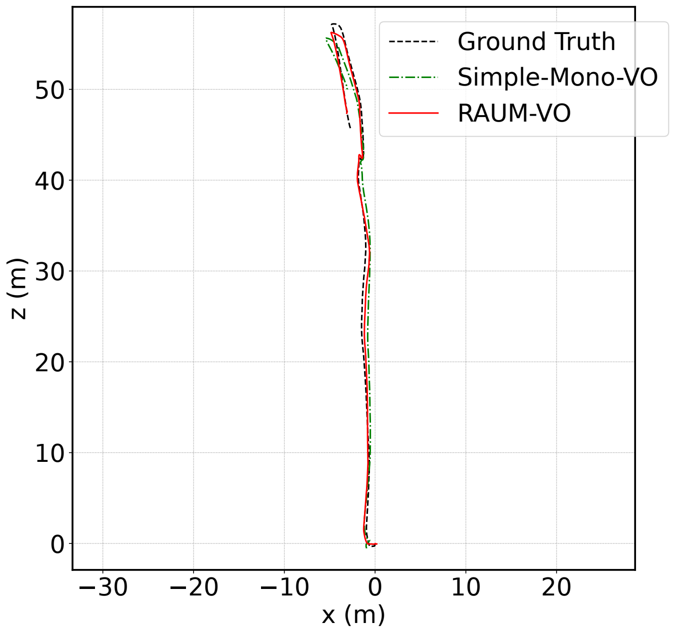


(b) Seq. 05

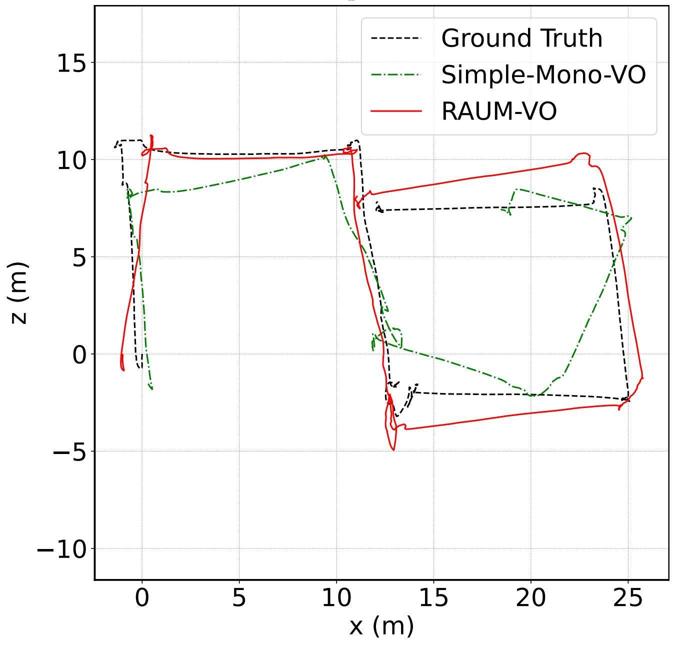


(c) Seq. 06

Figure 5.19: Belval train trajectories



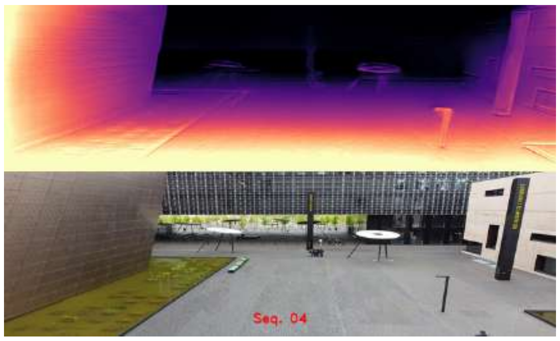
(a) Seq. 03



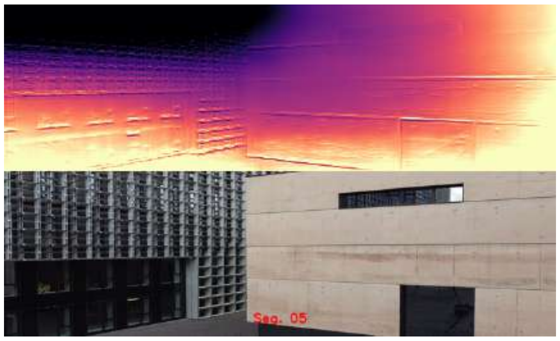
(b) Seq. 07

Figure 5.20: Belval test trajectories

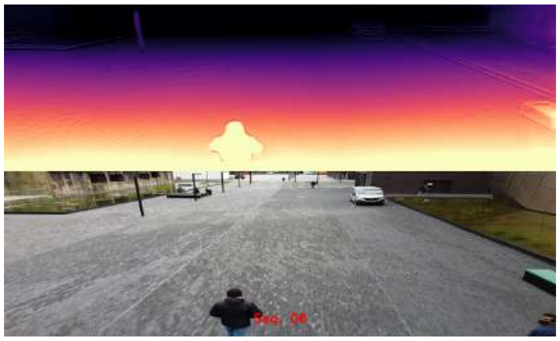
Depth Estimation



(a) Seq. 04



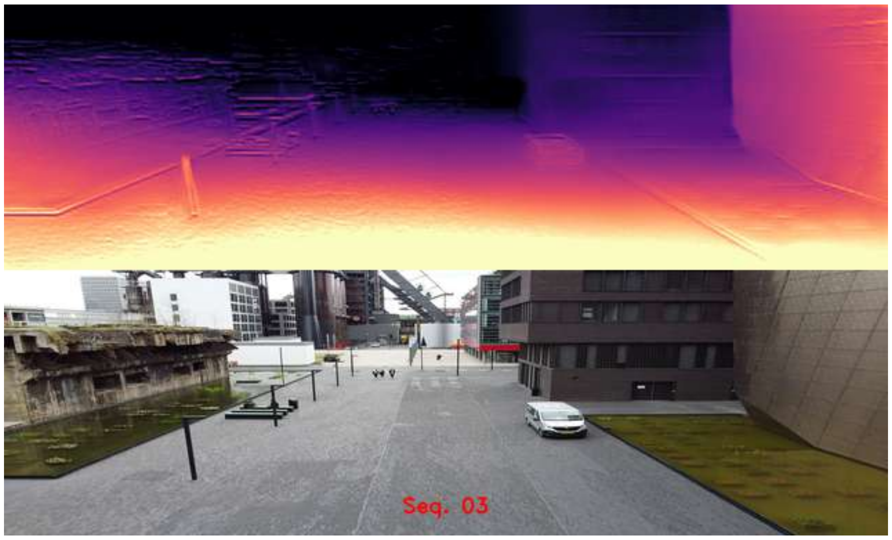
(b) Seq. 05



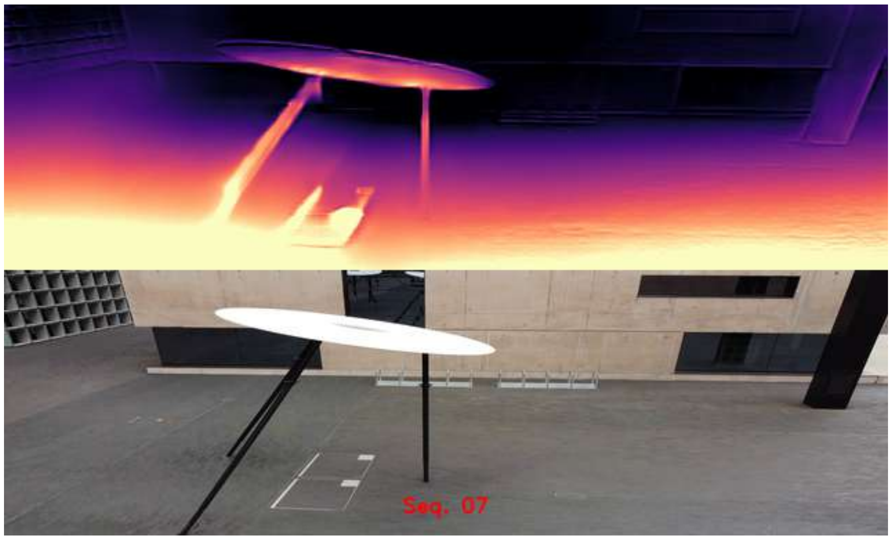
(c) Seq. 06

Figure 5.21: Example of estimated depth for Belval train sequences

The figure shows the depth maps estimated for the first frame of each train sequence in the Belval dataset. The depth network can accurately describe the scene 3D structure of buildings, small static objects, and people in the camera FoV.



(a) Seq. 03



(b) Seq. 07

Figure 5.22: Example of estimated depth for Belval test sequences

The figure shows the depth maps estimated for the first frame of each test sequence in the Belval dataset. The depth maps present a few artefacts of objects apparently 'leaking" in the foreground and high-frequency details in correspondence of edges, which could be mitigated with a less extreme depth smoothness prior.

5.6 Summary

This chapter discussed detecting people from images and obtaining their position in 3D reconstructing an outdoor environment. In the context of this thesis, we presented a methodology to effectively and efficiently detect intruders or, in general, the presence of people in the sensible area, leading to the implementation of a fundamental component of an autonomous aerial surveillance system.

We analysed current approaches for a similar problem and proposed our solution for projecting 2D bounding boxes, obtained by training EfficientDet CNN for object detection on a virtually placed ground plane. We tuned a SotA SfM approach, *i.e.*, COLMAP, to reconstruct a 3D scene from a sparse collection of images and aligned it with GPS coordinates to obtain the metric scale. Hence, after placing a ground plane in the environment, we locate the people in the 3D space using the camera to world transformation obtained with the global localisation network. Therefore, we presented the results on the AU-AIR dataset for outdoor object detection, providing a multi-modal sensor data recorder from a drone.

Furthermore, we introduced a novel outdoor environment created from images collected with a simple commercial drone flying in a populated urban area. Hence, we use the SfM approach to obtain the ground truth labels necessary to evaluate our proposed RAUM-VO method for ego-motion estimation. Finally, we concluded the chapter with an analysis of the results on the novel dataset, showing that our proposed method outperforms traditional V-SLAM in outdoor environments with repetitive textures and from an UAV performing purely rotational motions.

6
Conclusion

In this thesis, we addressed the application task of video surveillance that we intended to carry out autonomously with the support of an aerial robot, *i.e.*, a UAV. Thus, after analysing the complexity of an autonomous system, we focused our attention on the perceptual level of the UAV, which is a core aspect in the context of the treated application challenge. In this regard, we defined an autonomous aerial surveillance mission as enabling an UAV to autonomously navigate in a controlled urban environment, where the drone has to detect people who are possibly unauthorised to access a specific area around buildings and may be considered intruders. Therefore, other than autonomous control and planning, which are not treated in this work, an intelligent robotic agent needs solutions for answering the following perception-related questions to complete the mission successfully: *"Where am I located?"*, *"What is the structure of the surrounding environment?"*, and *"Which objects lie in my field of perception and where?"*. These questions constitute the general objective of the thesis, and we declined them into practical problems.

While the perception may rely on a broad spectrum of sensors, we restricted the hardware platform to small commercial drones that can fly in urban areas without any regulation restrictions, *i.e.*, the newly introduced EASA rules. Primarily, we targeted algorithms relying on a limited set of sensors carried by light-weight quadrotors, possibly less than 250 grams, such as the Parrot Bepop or the DJI Mavic Mini 2 used in our last experiments. Therefore, we focused on monocular cameras, which are the most common in this type of drone, and on computer vision algorithms for finding solutions to the posed challenges. Moreover, we noticed those leading robotic industries, such as Tesla, are moving towards the direction of purely vision systems for developing robots with autonomous capabilities, motivating, even more, our research in monocular vision. Additionally, we pushed further the restrictions on the sensor by excluding the IMU, which is rarely synchronised in commercial platforms and would require additional considerations for proper integration with the vision output. Instead, the GPS has been applied to obtain maps that are scaled and aligned with the geographic coordinates of the scene, which is the sole information that is not retrievable with monocular cameras. Finally, due to the demanding nature of the active computer vision topics, we identified DL as the general paradigm for establishing a valid methodology in most cases. For this reason, we described the theoretical framework of DL in chapter 2 and, especially, of the CNN, which is the preferred neural model for image processing.

Contributions

The contributions of this thesis revolve around the perception questions recalled above. The primary problem has been the UAV self-localisation because it is deemed fundamental for navigating the controlled environment and contributes to discovering the position of the people that are possibly detected. Then, distinguishing between the ego-motion estimation, or odometry, and global localisation, which is performed relative to an established map or reconstruction of the environment, we considered the two problems separately.

Regarding the ego-motion estimation, in chapter 3, we presented RAUM-VO, an algorithm to improve the monocular odometry estimates of unsupervised pose networks. To this end, we introduced an additional self-supervision loss using a SotA algorithm, named for simplicity F2F, for estimating the rotation between two camera frames utilising a set of local features. Consequently, we adjusted the rotation predicted by the trained pose network using the motion estimated by F2F during online inference to improve the final odometry. Finally, we compared our method with SotA approaches on the widely adopted KITTI benchmark. RAUM-VO enhances the performance of pose networks and is comparably good to more complex hybrid methods while being more straightforward and efficient to implement. Furthermore, the rotation adjustment procedure reduces the rotational drift, which is the more challenging component of the pose to learn. Also, RAUM-VO can estimate more accurate poses on trained scenes than traditional SLAM methods, such as ORB-SLAM.

- These findings have been published in the *Sensors* journal [48].

Regarding global localisation, we reviewed the literature on VBL, and we identified, among the direct pose estimations paradigms, the end-to-end learning approach as the more efficient and well-suited to our objective. Therefore, we implemented a neural network model based on the MobileNetV2 architecture, named MobilePoseNet, to hallucinate abstract features from a monocular RGB camera image. Consequently, these are processed by an MLP for regressing a global pose vector. Moreover, we compared multiple design options for the MLP performing an ablation study. From the experimental result on two environments with opposite characteristics, one indoor and one outdoor, we noted a monolithic MLP design, *i.e.*, sharing the weights between translation and rotation regression layers, outperforms the others on smaller scenarios. Instead, separating the last regression layers for outdoor environments obtained slightly better results. Finally, we evaluated the run-time performance on an NVIDIA TX2 and showed that the MobilePoseNet runs faster than the SotA PoseNet while retaining its general capabilities.

- These findings have been published in the proceedings of the *International Conference on Computer Analysis of Images and Patterns* [50].

Related to global localisation and in conjunction with the next object detection topic, we investigated the impact of dynamic objects on the training of global pose regressors. To this end, we performed an ablation study with multiple trained networks and elements. In detail, we introduced a dataset pre-processing step to mask dynamic objects before training a global localisation CNN model. Comparing the localisation errors, we showed that the presence of objects might cause only a minor performance degradation proportional to the percentage of occluded pixels. Furthermore, we implemented techniques to compute gradient saliency maps and visualised the image features on which the network focuses to regress the pose. Remarkably, the maps highlight pixels around the building's structural elements more, but static objects, such as vehicles, may cause the network to overfit to produce the pose prediction.

- These findings have been published in the proceedings of the *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)* [49].

Following, we tackled the second identified problem in the context of the surveillance application, *i.e.*, the detection of people and the estimation of their location in the same navigation environment. Notably, this has been placed in a subordinate position to understanding the robot's pose since we remarked on the importance of localisation as a fundamental component in an autonomous robotic system.

Hence, we discussed the details of an EfficientDet architecture for detecting people and of the Focal loss for addressing the imbalance class issue caused by our particular use case of binary object classification. Additionally, we deployed the network to be trained on a server cluster and exploited the available multi-GPU hardware leveraging distributed computation techniques. Following, we derived the equations to find the 3D position of the detected people in a reconstructed environment. For this purpose, we proposed to find the intersection point between a line passing by an object 2D bounding box base central point and an estimated ground plane. Ultimately, we used the COLMAP SotA implementation of an incremental SfM to create the 3D model of the environment necessary for the people 3D localisation algorithm. Additionally, we aligned the reconstructed scene with the approximate recorded GPS coordinates of the registered camera frames to obtain the correct metric scale. Furthermore, we introduced a novel dataset with images recorded from a DJI Mini 2 drone and its monocular camera flying on the University of Luxembourg's Belval campus. We concluded with extensive results to prove the validity of the methodologies developed throughout this thesis on natural outdoor urban environments.

- These findings have not been published yet, but will be submitted for review to a journal. However, related articles have been published in collaboration with Cazzato *et al.*. in the *Journal of Imaging* [33], in the proceedings of the *15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications* [35], and in the proceeding of the *International Joint Conference on Computer Vision, Imaging and Computer Graphics* [34].

Other works related to the perception system, as described in section 1.1, have been published in collaboration with other researchers:

- on the human recognition from ocular biometrics in the [36] *Proceedings of the 2019 3rd International Conference on Artificial Intelligence and Virtual Reality*, and in *Applied Sciences* [31];
- on the human-robot interaction *Proceedings of the 2019 3rd International Conference on Artificial Intelligence and Virtual Reality* [32].

Future Work

Regarding localisation, newly appeared DL techniques could better exploit the local future extracted by the neural network, such as Superpoint, for the ego-motion estimation and predict the global pose. Remarkably, the Transformers networks have revolutionised the field of NLP. Still, a great emphasis has been put on applying them to computer vision problems, such as object detection and image captioning. We believe that their capabilities of finding patterns by posing attention to a different part of the image while retaining the information of spatial pixel distance may ease the problem of finding the transformation matrix on which localisation is standing. Furthermore, this technique would combine the relative and global pose estimation problems with an architecture that maximises parameter sharing and computation reuse. Finally, a similar approach between more related tasks, *e.g.*, ego-motion and global pose estimation, lets us foresee more promising results.

References

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., & Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[2] Abdulla, W. (2017). Mask r-cnn for object detection and instance segmentation on keras and tensorflow. https://github.com/matterport/Mask_RCNN.

[3] Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

[4] Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., Curless, B., Seitz, S. M., & Szeliski, R. (2011). Building rome in a day. *Communications of the ACM*, 54(10), 105–112.

[5] Agarwal, S., Mierle, K., & Team, T. C. S. (2022). Ceres Solver. https://github.com/ceres-solver/ceres-solver.

[6] Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Ho, D., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jang, E., Ruano, R. J., Jeffrey, K., Jesmonth, S., Joshi, N., Julian, R., Kalashnikov, D., Kuang, Y., Lee, K.-H., Levine, S., Lu, Y., Luu, L., Parada, C., Pastor, P., Quiambao, J., Rao, K., Rettinghouse, J., Reyes, D., Sermanet, P., Sievers, N., Tan, C., Toshev, A., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Xu, S., & Yan, M. (2022). Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2022.00000*.

[7] Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep

learning: Concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8(1), 1–74.

[8] Araar, O. & Aouf, N. (2014). A new hybrid approach for the visual servoing of vtol uavs from unknown geometries. In *22nd Mediterranean Conference on Control and Automation* (pp. 1425–1432).: IEEE.

[9] Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., & Sivic, J. (2016). Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5297–5307).

[10] Babin, P., Giguere, P., & Pomerleau, F. (2019). Analysis of robust functions for registration algorithms. In *2019 International Conference on Robotics and Automation (ICRA)* (pp. 1451–1457).: IEEE.

[11] Balamurugan, G., Valarmathi, J., & Naidu, V. (2016). Survey on uav navigation in gps denied environments. In *2016 International conference on signal processing, communication, power and embedded system (SCOPES)* (pp. 198–204).: IEEE.

[12] Balntas, V., Riba, E., Ponsa, D., & Mikolajczyk, K. (2016). Learning local feature descriptors with triplets and shallow convolutional neural networks. *British Machine Vision Conference 2016, BMVC 2016*, 2016-September, 119.1–119.11.

[13] Bavle, H., Sanchez-Lopez, J. L., Schmidt, E. F., & Voos, H. (2021). From slam to situational awareness: Challenges and survey. *arXiv preprint arXiv:2110.00273*.

[14] Berg, A., Deng, J., & Fei-Fei, L. (2010). Large scale visual recognition challenge 2010.

[15] Bescos, B., Fácil, J. M., Civera, J., & Neira, J. (2018). Dynaslam: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters*, 3(4), 4076–4083.

[16] Bian, J., Li, Z., Wang, N., Zhan, H., Shen, C., Cheng, M., & Reid, I. D. (2019). Unsupervised scale-consistent depth and ego-motion learning from monocular video. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada* (pp. 35–45).

[17] Bian, J.-W., Zhan, H., Wang, N., Li, Z., Zhang, L., Shen, C., Cheng, M.-M., & Reid, I. (2021). Unsupervised scale-consistent depth learning from video. *International Journal of Computer Vision*, 129(9), 2548–2564.

[18] Bleyer, M., Rhemann, C., & Rother, C. (2011). Patchmatch stereo-stereo matching with slanted support windows. In *Bmvc*, volume 11 (pp. 1–11).

[19] Bloesch, M., Czarnowski, J., Clark, R., Leutenegger, S., & Davison, A. J. (2018). CodeSLAM - Learning a Compact, Optimisable Representation for Dense Visual SLAM. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (pp. 2560–2568).

[20] Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

[21] Bozcan, I. & Kayacan, E. (2020). Au-air: A multi-modal unmanned aerial vehicle dataset for low altitude traffic surveillance.

[22] Brahmbhatt, S., Gu, J., Kim, K., Hays, J., & Kautz, J. (2018). Geometry-aware learning of maps for camera localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2616–2625).

[23] Bruno, H. M. S. & Colombini, E. L. (2021). LIFT-SLAM: A deep-learning feature-based monocular visual SLAM method. *Neurocomputing*, 455, 97–110.

[24] Bustos, Á. P., Chin, T., Eriksson, A. P., & Reid, I. D. (2019). Visual SLAM: why bundle adjust? In *International Conference on Robotics and Automation, ICRA 2019, Montreal, QC, Canada, May 20-24, 2019* (pp. 2385–2391).: IEEE.

[25] C., D. H., Kim, K., Kannala, J., Pulli, K., & Heikkilä, J. (2014). DT-SLAM: deferred triangulation for robust SLAM. In *2nd International Conference on 3D Vision, 3DV 2014, Tokyo, Japan, December 8-11, 2014, Volume 1* (pp. 609–616).: IEEE Computer Society.

[26] Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., & Leonard, J. J. (2016). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6), 1309–1332.

[27] Cantzler, H. (1981). Random sample consensus (ransac). *Institute for Perception, Action and Behaviour, Division of Informatics, University of Edinburgh*.

[28] Cao, Z., Simon, T., Wei, S.-E., & Sheikh, Y. (2017). Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7291–7299).

[29] Carlone, L., Tron, R., Daniilidis, K., & Dellaert, F. (2015). Initialization techniques for 3d SLAM: A survey on rotation estimation and its use in pose graph optimization. In *IEEE International Conference on Robotics and Automation, ICRA 2015, Seattle, WA, USA, 26-30 May, 2015* (pp. 4597–4604).: IEEE.

[30] Casser, V., Pirk, S., Mahjourian, R., & Angelova, A. (2019). Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33 (pp. 8001–8008).

[31] Cazzato, D., Carcagnì, P., Cimarelli, C., Voos, H., Distante, C., & Leo, M. (2020a). Ocular biometrics recognition by analyzing human exploration during video observations. *Applied Sciences*, 10(13), 4548.

[32] Cazzato, D., Cimarelli, C., Sanchez-Lopez, J. L., Olivares-Mendez, M. A., & Voos, H. (2019a). Real-time human head imitation for humanoid robots. In *Proceedings of the 2019 3rd International Conference on Artificial Intelligence and Virtual Reality* (pp. 65–69).

[33] Cazzato, D., Cimarelli, C., Sanchez-Lopez, J. L., Voos, H., & Leo, M. (2020b). A survey of computer vision methods for 2d object detection from unmanned aerial vehicles. *Journal of Imaging*, 6(8), 78.

[34] Cazzato, D., Cimarelli, C., & Voos, H. (2020c). On-board uav pilots identification in counter uav images. In *International Joint Conference on Computer Vision, Imaging and Computer Graphics* (pp. 414–430).: Springer.

[35] Cazzato, D., Cimarelli, C., & Voos, H. (2020d). A preliminary study on the automatic visual based identification of uav pilots from counter uavs. In *15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications-Volume 5: VISAPP, Valletta 27-29 February 2020* (pp. 582–589).

[36] Cazzato, D., Leo, M., Carcagnì, P., Cimarelli, C., & Voos, H. (2019b). Understanding and modelling human attention for soft biometrics purposes. In *Proceedings of the 2019 3rd International Conference on Artificial Intelligence and Virtual Reality* (pp. 51–55).

[37] Chao, H., Cao, Y., & Chen, Y. (2010). Autopilots for small unmanned aerial vehicles: a survey. *International Journal of Control, Automation and Systems*, 8(1), 36–44.

[38] Chatterjee, A. & Govindu, V. M. (2013). Efficient and robust large-scale rotation averaging. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013* (pp. 521–528).: IEEE Computer Society.

[39] Chatterjee, A. & Govindu, V. M. (2018). Robust relative rotation averaging. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4), 958–972.

[40] Chen, C., Zhu, H., Li, M., & You, S. (2018). A review of visual-inertial simultaneous localization and mapping from filtering-based and optimization-based perspectives. *Robotics*, 7(3), 45.

[41] Chen, H., Wang, X.-m., & Li, Y. (2009). A survey of autonomous control for uav. In *2009 International Conference on Artificial Intelligence and Computational Intelligence*, volume 2 (pp. 267–271).: IEEE.

[42] Chen, W., Liu, Y., Wang, W., Bakker, E. M., Georgiou, T., Fieguth, P., Liu, L., & Lew, M. (2021). Deep image retrieval: A survey. *ArXiv*.

[43] Chen, Y., Schmid, C., & Sminchisescu, C. (2019). Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 7063–7072).

[44] Cheng, R., Agia, C., Meger, D., & Dudek, G. (2020). Depth prediction for monocular direct visual odometry. In *2020 17th Conference on Computer and Robot Vision (CRV)* (pp. 70–77).: IEEE Computer Society.

[45] Chetverikov, D., Svirko, D., Stepanov, D., & Krsek, P. (2002). The trimmed iterative closest point algorithm. In *16th International Conference on Pattern Recognition, ICPR 2002, Quebec, Canada, August 11-15, 2002* (pp. 545–548).: IEEE Computer Society.

[46] Chng, C., Parra, Á., Chin, T., & Latif, Y. (2020). Monocular rotational odometry with incremental rotation averaging and loop closure. *CoRR*, abs/2010.01872.

[47] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1251–1258).

[48] Cimarelli, C., Bavle, H., Sanchez-Lopez, J. L., & Voos, H. (2022). Raum-vo: Rotational adjusted unsupervised monocular visual odometry. *Sensors*, 22(7), 2651.

[49] Cimarelli, C., Cazzato, D., Olivares-Mendez, M. A., & Voos, H. (2019a). A case study on the impact of masking moving objects on the camera pose regression with cnns. In *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)* (pp. 1–8).: IEEE.

[50] Cimarelli, C., Cazzato, D., Olivares-Mendez, M. A., & Voos, H. (2019b). Faster visual-based localization with mobile-posenet. In *International Conference on Computer Analysis of Images and Patterns* (pp. 219–230).: Springer.

[51] Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

[52] Cummins, M. & Newman, P. (2008). Fab-map: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6), 647–665.

[53] Czarnowski, J., Laidlow, T., Clark, R., & Davison, A. J. (2020). DeepFactors: Real-Time Probabilistic Dense Monocular SLAM. *IEEE Robotics and Automation Letters*, 5(2), 721–728.

[54] Dai, X., Zhang, P., Wu, B., Yin, H., Sun, F., Wang, Y., Dukhan, M., Hu, Y., Wu, Y., Jia, Y., et al. (2019). Chamnet: Towards efficient network design through platform-aware model adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 11398–11407).

[55] Dalal, N. & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1 (pp. 886–893).: IEEE.

[56] Davidovich, B., Nassi, B., & Elovici, Y. (2022). Towards the detection of gps spoofing attacks against drones by analyzing camera's video stream. *Sensors*, 22(7), 2608.

[57] Davison, A. J., Reid, I. D., Molton, N., & Stasse, O. (2007). Monoslam: Real-time single camera SLAM. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6), 1052–1067.

[58] Dellaert, F. (2012). *Factor graphs and GTSAM: A hands-on introduction*. Technical report, Georgia Institute of Technology.

[59] Dellaert, F. (2021). Factor graphs: Exploiting structure in robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 4, 141–166.

[60] Dellaert, F. & Kaess, M. (2006). Square root SAM: simultaneous localization and mapping via square root information smoothing. *Int. J. Robotics Res.*, 25(12), 1181–1203.

[61] Dellaert, F. & Kaess, M. (2017). Factor graphs for robot perception. *Found. Trends Robotics*, 6(1-2), 1–139.

[62] Dellaert, F., Rosen, D. M., Wu, J., Mahony, R. E., & Carlone, L. (2020). Shonan rotation averaging: Global optimality by surfing so(p)^n. In A. Vedaldi, H. Bischof, T. Brox, & J. Frahm (Eds.), *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part VI*, volume 12351 of *Lecture Notes in Computer Science* (pp. 292–308).: Springer.

[63] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009a). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248–255).: Ieee.

[64] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009b). Imagenet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 248–255).

[65] DeTone, D., Malisiewicz, T., & Rabinovich, A. (2018a). Self-improving visual odometry. *arXiv preprint arXiv:1812.03245*.

[66] DeTone, D., Malisiewicz, T., & Rabinovich, A. (2018b). Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 224–236).

[67] DeVries, T. & Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*.

[68] Dhp1080 (2019). Diagram of a neuron. https://commons.wikimedia.org/wiki/File:Neuron.svg. Accessed: 2022-04-20, CC BY-SA 3.0.

[69] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., & Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning* (pp. 647–655).

[70] Dumoulin, V. & Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*.

[71] Durrant-Whyte, H. & Bailey, T. (2006). Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2), 99–110.

[72] Dusmanu, M., Rocco, I., Pajdla, T., Pollefeys, M., Sivic, J., Torii, A., & Sattler, T. (2019). D2-net: A trainable CNN for joint description and detection of local features. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2019-June (pp. 8084–8093).

[73] Elmokadem, T. & Savkin, A. V. (2021). Towards fully autonomous uavs: A survey. *Sensors*, 21(18), 6223.

[74] Elsken, T., Metzen, J. H., & Hutter, F. (2018). Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377*.

[75] Endsley, M. R. (1995). Toward a theory of situation awareness in dynamic systems. *Hum. Factors*, 37(1), 32–64.

[76] Engel, J., Koltun, V., & Cremers, D. (2018). Direct sparse odometry. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(3), 611–625.

[77] Engel, J., Schöps, T., & Cremers, D. (2014). LSD-SLAM: large-scale direct monocular SLAM. In D. J. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II*, volume 8690 of *Lecture Notes in Computer Science* (pp. 834–849).: Springer.

[78] Engel, J., Sturm, J., & Cremers, D. (2013). Semi-dense visual odometry for a monocular camera. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013* (pp. 1449–1456).: IEEE Computer Society.

[79] Eriksson, A. P., Olsson, C., Kahl, F., & Chin, T. (2021). Rotation averaging with the chordal distance: Global minimizers and strong duality. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(1), 256–268.

[80] Felzenszwalb, P., McAllester, D., & Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model. In *2008 IEEE conference on computer vision and pattern recognition* (pp. 1–8).: Ieee.

[81] Felzenszwalb, P. F., Girshick, R. B., & McAllester, D. (2010a). Cascade object detection with deformable part models. In *2010 IEEE Computer society conference on computer vision and pattern recognition* (pp. 2241–2248).: Ieee.

[82] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010b). Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9), 1627–1645.

[83] Florian Fuchs (2021). Dji mini 2. `https://commons.wikimedia.org/wiki/File:DJI_Mini_2.jpg`. Accessed: 2022-04-20, CC BY-SA 4.0.

[84] Forster, C., Pizzoli, M., & Scaramuzza, D. (2014). SVO: fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014* (pp. 15–22).: IEEE.

[85] Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780), 1612.

[86] Gálvez-López, D. & Tardos, J. D. (2012). Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5), 1188–1197.

[87] Gao, X. & Zhang, T. (2021). *Introduction to Visual SLAM: From Theory to Practice*. Springer Nature.

[88] Garg, R., Kumar, B. G. V., Carneiro, G., & Reid, I. D. (2016). Unsupervised CNN for single view depth estimation: Geometry to the rescue. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII*, volume 9912 of *Lecture Notes in Computer Science* (pp. 740–756).: Springer.

[89] Garg, S., Fischer, T., & Milford, M. (2021). Where is your place, visual place recognition? *arXiv preprint arXiv:2103.06443*.

[90] Gauglitz, S., Sweeney, C., Ventura, J., Turk, M. A., & Höllerer, T. (2012). Live tracking and mapping from both general and rotation-only camera motion. In *11th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2012, Atlanta, GA, USA, November 5-8, 2012* (pp. 13–22).: IEEE Computer Society.

[91] Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11), 1231–1237.

[92] Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition* (pp. 3354–3361).: IEEE.

[93] Geiger, A., Ziegler, J., & Stiller, C. (2011). Stereoscan: Dense 3d reconstruction in real-time. In *IEEE Intelligent Vehicles Symposium (IV), 2011, Baden-Baden, Germany, June 5-9, 2011* (pp. 963–968).: IEEE.

[94] Ghiasi, G., Lin, T.-Y., & Le, Q. V. (2019). Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 7036–7045).

[95] Ghosh, S., Das, N., Das, I., & Maulik, U. (2019). Understanding deep learning techniques for image segmentation. *ACM Computing Surveys (CSUR)*, 52(4), 1–35.

[96] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440–1448).

[97] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580–587).

[98] Girshick, R., Felzenszwalb, P., & McAllester, D. (2011). Object detection with grammar models. *Advances in neural information processing systems*, 24.

[99] Godard, C., Aodha, O. M., & Brostow, G. J. (2017). Unsupervised monocular depth estimation with left-right consistency. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017* (pp. 6602–6611).: IEEE Computer Society.

[100] Godard, C., Aodha, O. M., Firman, M., & Brostow, G. J. (2019). Digging into self-supervised monocular depth estimation. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019* (pp. 3827–3837).: IEEE.

[101] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

[102] Gordo, A., Almazán, J., Revaud, J., & Larlus, D. (2016). Deep image retrieval: Learning global representations for image search. In *European conference on computer vision* (pp. 241–257).: Springer.

[103] Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., & He, K. (2017). Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*.

[104] Grisetti, G., Guadagnino, T., Aloise, I., Colosi, M., Della Corte, B., & Schlegel, D. (2020). Least squares optimization: From theory to practice. *Robotics*, 9(3), 51.

[105] Grisetti, G., Kümmerle, R., Stachniss, C., & Burgard, W. (2010). A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4), 31–43.

[106] Guizilini, V., Ambrus, R., Pillai, S., Raventos, A., & Gaidon, A. (2020). 3d packing for self-supervised monocular depth estimation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020* (pp. 2482–2491).: Computer Vision Foundation / IEEE.

[107] Harltey, A. & Zisserman, A. (2006). *Multiple view geometry in computer vision (2. ed.).* Cambridge University Press.

[108] Hartley, R. I., Aftab, K., & Trumpf, J. (2011). L1 rotation averaging using the weiszfeld algorithm. In *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011* (pp. 3041–3048).: IEEE Computer Society.

[109] Hartley, R. I., Trumpf, J., Dai, Y., & Li, H. (2013). Rotation averaging. *Int. J. Comput. Vis.*, 103(3), 267–305.

[110] Hayat, S., Jung, R., Hellwagner, H., Bettstetter, C., Emini, D., & Schnieders, D. (2021). Edge computing in 5g for drone navigation: What to offload? *IEEE Robotics and Automation Letters*, 6(2), 2571–2578.

[111] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 2961–2969).

[112] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1026–1034).

[113] He, K., Zhang, X., Ren, S., & Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

[114] He, K., Zhang, X., Ren, S., & Sun, J. (2016b). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

[115] He, K., Zhang, X., Ren, S., & Sun, J. (2016c). Identity mappings in deep residual networks. In *European conference on computer vision* (pp. 630–645).: Springer.

[116] Hirschberg, J. & Manning, C. D. (2015). Advances in natural language processing. *Science*, 349(6245), 261–266.

[117] Hochreiter, S. & Schmidhuber, J. (1997a). Flat minima. *Neural computation*, 9(1), 1–42.

[118] Hochreiter, S. & Schmidhuber, J. (1997b). Long short-term memory. *Neural computation*, 9(8), 1735–1780.

[119] Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5), 359–366.

[120] Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. (2019). Searching for mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1314–1324).

[121] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[122] Huang, H.-M. (2007). Autonomy levels for unmanned systems (alfus) framework: safety and application issues. In *Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems* (pp. 48–53).

[123] Huynh, D. Q. (2009). Metrics for 3d rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2), 155–164.

[124] Ioffe, S. & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

[125] Iqbal, H. (2020). Plotneuralnet. `https://github.com/HarisIqbal88/PlotNeuralNet`. GitHub repository.

[126] Irschara, A., Zach, C., Frahm, J.-M., & Bischof, H. (2009). From structure-from-motion point clouds to fast location recognition. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2599–2606).: IEEE.

[127] Jaderberg, M., Simonyan, K., Zisserman, A., & Kavukcuoglu, K. (2015). Spatial transformer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada* (pp. 2017–2025).

[128] Jégou, H., Douze, M., & Schmid, C. (2009). On the burstiness of visual elements. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 1169–1176).: IEEE.

[129] Jiang, H., Ding, L., Sun, Z., & Huang, R. (2021). Unsupervised monocular depth perception: Focusing on moving objects. *IEEE Sensors Journal*, 21(24), 27225–27237.

[130] Johannes L. Schoenberger (2022). Colmap pipeline diagram. https://colmap.github.io/tutorial.html#id21. Accessed: 2022-04-20.

[131] Josephson, K. & Byrod, M. (2009). Pose estimation with radial distortion and unknown focal length. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2419–2426).: IEEE.

[132] Kabsch, W. (1976). A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5), 922–923.

[133] Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., & Dellaert, F. (2012). isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2), 216–235.

[134] Kang, R., Shi, J., Li, X., Liu, Y., & Liu, X. (2019). Df-slam: A deep-learning enhanced visual slam system based on deep local features. *arXiv preprint arXiv:1901.07223*.

[135] Kendall, A. & Cipolla, R. (2017). Geometric loss functions for camera pose regression with deep learning. In *Camera relocalization by computing pairwise relative poses using convolutional neural network* (pp. 5974–5983).

[136] Kendall, A., Gal, Y., & Cipolla, R. (2018). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 7482–7491).

[137] Kendall, A., Grimes, M., & Cipolla, R. (2015). Posenet: A convolutional network for real-time 6-dof camera relocalization. In *IEEE International Conference on Computer Vision*.

[138] Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

[139] Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[140] Kingma, D. P. & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[141] KlausFoehl (2018). Parrot anafi drone. `https://commons.wikimedia.org/wiki/File:Parrot_Anafi_Drone01_2018-07-19.jpg`. Accessed: 2022-04-20, CC BY-SA 4.0.

[142] Klein, G. & Murray, D. W. (2007). Parallel tracking and mapping for small AR workspaces. In *Sixth IEEE/ACM International Symposium on Mixed and Augmented Reality, ISMAR 2007, 13-16 November 2007, Nara, Japan* (pp. 225–234).: IEEE Computer Society.

[143] Kneip, L. & Furgale, P. (2014). Opengv: A unified and generalized approach to real-time calibrated geometric vision. In *2014 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1–8).: IEEE.

[144] Kneip, L. & Lynen, S. (2013). Direct optimization of frame-to-frame rotation. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013* (pp. 2352–2359).: IEEE Computer Society.

[145] Kneip, L., Siegwart, R., & Pollefeys, M. (2012). Finding the exact rotation between two images independently of the translation. In A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, & C. Schmid (Eds.), *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI*, volume 7577 of *Lecture Notes in Computer Science* (pp. 696–709).: Springer.

[146] Kolmet, M., Zhou, Q., Osep, A., & Leal-Taixe, L. (2022). Text2pos: Text-to-point-cloud cross-modal localization. *arXiv preprint arXiv:2203.15125*.

[147] Krizhevsky, A. & Hinton, G. (2010). Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40(7).

[148] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

[149] Kschischang, F., Frey, B., & Loeliger, H.-A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2), 498–519.

[150] Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., & Burgard, W. (2011). g 2 o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation* (pp. 3607–3613).: IEEE.

[151] Kümmerle, R., Steder, B., Dornhege, C., Ruhnke, M., Grisetti, G., Stachniss, C., & Kleiner, A. (2009). On measuring the accuracy of slam algorithms. *Autonomous Robots*, 27(4), 387–407.

[152] Kutyniok, G. (2022). The mathematics of artificial intelligence. *arXiv preprint arXiv:2203.08890*.

[153] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, 521(7553), 436–444.

[154] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.

[155] LeCun, Y., Kavukcuoglu, K., & Farabet, C. (2010). Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems* (pp. 253–256).

[156] Lee, J. M. (2013). Smooth manifolds. In *Introduction to Smooth Manifolds* (pp. 1–31). Springer.

[157] Lee, S., Im, S., Lin, S., & Kweon, I. S. (2019). Learning residual flow as dynamic motion from stereo videos. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1180–1186).: IEEE.

[158] Lee, S. H. & Civera, J. (2021). Rotation-only bundle adjustment. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021* (pp. 424–433).: Computer Vision Foundation / IEEE.

[159] Lee, S. J., Kim, D., Hwang, S. S., & Lee, D. (2021). Local to global: Efficient visual localization for a monocular camera. In *Proceedings - 2021 IEEE Winter Conference on Applications of Computer Vision, WACV 2021* (pp. 2230–2239).

[160] Lepetit, V., Moreno-Noguer, F., & Fua, P. (2009). Ep*n*p: An accurate $O(n)$ solution to the p*n*p problem. *Int. J. Comput. Vis.*, 81(2), 155–166.

[161] Lesort, T., Lomonaco, V., Stoian, A., Maltoni, D., Filliat, D., & Díaz-Rodríguez, N. (2020). Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information fusion*, 58, 52–68.

[162] Leutenegger, S., Furgale, P., Rabaud, V., Chli, M., Konolige, K., & Siegwart, R. (2013). Keyframe-based visual-inertial slam using nonlinear optimization. *Proceedings of Robotis Science and Systems (RSS) 2013*.

[163] Li, D., Shi, X., Long, Q., Liu, S., Yang, W., Wang, F., Wei, Q., & Qiao, F. (2020a). DXSLAM: A robust and efficient visual SLAM system with deep features. In *IEEE International Conference on Intelligent Robots and Systems* (pp. 4958–4965).

[164] Li, H., Gordon, A., Zhao, H., Casser, V., & Angelova, A. (2020b). Unsupervised monocular depth learning in dynamic scenes. *arXiv preprint arXiv:2010.16404*.

[165] Li, H., Xu, Z., Taylor, G., Studer, C., & Goldstein, T. (2018a). Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31.

[166] Li, R., Wang, S., Long, Z., & Gu, D. (2018b). Undeepvo: Monocular visual odometry through unsupervised deep learning. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018* (pp. 7286–7291).: IEEE.

[167] Li, S., Wu, X., Cao, Y., & Zha, H. (2021). Generalizing to the open world: Deep visual odometry with online adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021* (pp. 13184–13193).: Computer Vision Foundation / IEEE.

[168] Li, Y., Snavely, N., & Huttenlocher, D. P. (2010). Location recognition using prioritized feature matching. In *European conference on computer vision* (pp. 791–804).: Springer.

[169] Li, Y., Ushiku, Y., & Harada, T. (2019). Pose graph optimization for unsupervised monocular visual odometry. In *2019 International Conference on Robotics and Automation (ICRA)* (pp. 5439–5445).: IEEE.

[170] Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y., & Sun, J. (2018c). Detnet: A backbone network for object detection. *arXiv preprint arXiv:1804.06215*.

[171] Liew, C. F., DeLatte, D., Takeishi, N., & Yairi, T. (2017). Recent developments in aerial robotics: A survey and prototypes overview. *arXiv preprint arXiv:1711.10085*.

[172] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017a). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2117–2125).

[173] Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017b). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision* (pp. 2980–2988).

[174] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740–755).: Springer.

[175] Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., & Pietikäinen, M. (2020). Deep learning for generic object detection: A survey. *International journal of computer vision*, 128(2), 261–318.

[176] Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 8759–8768).

[177] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision* (pp. 21–37).: Springer.

[178] Loeliger, H.-A. (2004). An introduction to factor graphs. *IEEE Signal Processing Magazine*, 21(1), 28–41.

[179] Long, X., Deng, K., Wang, G., Zhang, Y., Dang, Q., Gao, Y., Shen, H., Ren, J., Han, S., Ding, E., et al. (2020). Pp-yolo: An effective and efficient implementation of object detector. *arXiv preprint arXiv:2007.12099*.

[180] Longuet-Higgins, H. C. (1981). A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828), 133–135.

[181] Loo, S. Y., Amiri, A. J., Mashohor, S., Tang, S. H., & Zhang, H. (2019a). CNN-SVO: Improving the mapping in semi-direct visual odometry using single-image depth prediction. *Proceedings - IEEE International Conference on Robotics and Automation*, 2019-May, 5218–5223.

[182] Loo, S. Y., Amiri, A. J., Mashohor, S., Tang, S. H., & Zhang, H. (2019b). Cnn-svo: Improving the mapping in semi-direct visual odometry using single-image depth prediction. In *2019 International Conference on Robotics and Automation (ICRA)* (pp. 5218–5223).: IEEE.

[183] Loshchilov, I. & Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.

[184] Loshchilov, I. & Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

[185] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2), 91–110.

[186] Lowry, S., Sünderhauf, N., Newman, P., Leonard, J. J., Cox, D., Corke, P., & Milford, M. J. (2015). Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1), 1–19.

[187] Lu, X., Li, Q., Li, B., & Yan, J. (2020). Mimicdet: Bridging the gap between one-stage and two-stage object detection. In *European Conference on Computer Vision* (pp. 541–557).: Springer.

[188] Lu, Y., Xue, Z., Xia, G.-S., & Zhang, L. (2018). A survey on vision-based uav navigation. *Geo-spatial information science*, 21(1), 21–32.

[189] Luo, C., Yang, Z., Wang, P., Wang, Y., Xu, W., Nevatia, R., & Yuille, A. L. (2020a). Every pixel counts ++: Joint learning of geometry and motion with 3d holistic understanding. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(10), 2624–2641.

[190] Luo, X., Huang, J., Szeliski, R., Matzen, K., & Kopf, J. (2020b). Consistent video depth estimation. *ACM Trans. Graph.*, 39(4), 71.

[191] Maas, A. L., Hannun, A. Y., Ng, A. Y., et al. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30 (pp.3̃).: Citeseer.

[192] Mahjourian, R., Wicke, M., & Angelova, A. (2018). Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018* (pp. 5667–5675).: Computer Vision Foundation / IEEE Computer Society.

[193] Maier, J. & Humenberger, M. (2013). Movement detection based on dense optical flow for unmanned aerial vehicles. *International Journal of Advanced Robotic Systems*, 10(2), 146.

[194] Martinec, D. & Pajdla, T. (2007). Robust rotation and translation estimation in multiview reconstruction. In *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), 18-23 June 2007, Minneapolis, Minnesota, USA*: IEEE Computer Society.

[195] Mascaro, R., Teixeira, L., Hinzmann, T., Siegwart, R., & Chli, M. (2018). GOMSF: Graph-Optimization Based Multi-Sensor Fusion for robust UAV Pose estimation. In *Proceedings - IEEE International Conference on Robotics and Automation* (pp. 1421–1428).: Institute of Electrical and Electronics Engineers Inc.

[196] Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., & Brox, T. (2016). A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4040–4048).

[197] McCulloch, W. S. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4), 115–133.

[198] Melekhov, I., Laskar, Z., Li, X., Wang, S., & Kannala, J. (2021). Digging into self-supervised learning of feature descriptors. In *2021 International Conference on 3D Vision (3DV)* (pp. 1144–1155).: IEEE.

[199] Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision* (pp. 405–421).: Springer.

[200] Ming, Y., Meng, X., Fan, C., & Yu, H. (2021). Deep learning for monocular depth estimation: A review. *Neurocomputing*, 438, 14–33.

[201] Mitchell, T. M. (1997). *Machine learning, International Edition*. McGraw-Hill Series in Computer Science. McGraw-Hill.

[202] Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2002). Fastslam: A factored solution to the simultaneous localization and mapping problem. In R. Dechter, M. J. Kearns, & R. S. Sutton (Eds.), *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence, July 28 - August 1, 2002, Edmonton, Alberta, Canada* (pp. 593–598).: AAAI Press / The MIT Press.

[203] Mur-Artal, R., Montiel, J. M. M., & Tardós, J. D. (2015). ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robotics*, 31(5), 1147–1163.

[204] Mur-Artal, R. & Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5), 1255–1262.

[205] Newcombe, R. A., Lovegrove, S. J., & Davison, A. J. (2011). Dtam: Dense tracking and mapping in real-time. In *2011 international conference on computer vision* (pp. 2320–2327).: IEEE.

[206] Nielsen, M. A. (2015). *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA.

[207] Nistér, D. (2004). An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(6), 756–777.

[208] Ono, Y., Trulls, E., Fua, P., & Yi, K. M. (2018). Lf-net: Learning local features from images. *Advances in neural information processing systems*, 31.

[209] Pachter, M. & Chandler, P. R. (1998). Challenges of autonomous control. *IEEE Control Systems Magazine*, 18(4), 92–97.

[210] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

[211] Pedoeem, J. & Huang, R. (2018). YOLO-LITE: A real-time object detection algorithm optimized for non-gpu computers. *CoRR*, abs/1811.05588.

[212] Pendleton, S. D., Andersen, H., Du, X., Shen, X., Meghjani, M., Eng, Y. H., Rus, D., & Ang, M. H. (2017). Perception, planning, control, and coordination for autonomous vehicles. *Machines*, 5(1), 6.

[213] Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., & Geiger, A. (2020). Convolutional occupancy networks. In *European Conference on Computer Vision* (pp. 523–540).: Springer.

[214] Perera, A. G., Al-Naji, A., Law, Y. W., & Chahl, J. (2018). Human detection and motion analysis from a quadrotor uav. In *IOP conference series: materials science and engineering*, volume 405 (pp. 012003).: IOP Publishing.

[215] Philbin, J., Chum, O., Isard, M., Sivic, J., & Zisserman, A. (2007). Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE conference on computer vision and pattern recognition* (pp. 1–8).: IEEE.

[216] Piasco, N., Sidibé, D., Demonceaux, C., & Gouet-Brunet, V. (2018). A survey on visual-based localization: On the benefit of heterogeneous data. *Pattern Recognition*, 74, 90–109.

[217] Pirchheim, C., Schmalstieg, D., & Reitmayr, G. (2013). Handling pure camera rotation in keyframe-based SLAM. In *IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2013, Adelaide, Australia, October 1-4, 2013* (pp. 229–238).: IEEE Computer Society.

[218] Popoola, O. P. & Wang, K. (2012). Video-based abnormal human behavior recognition—a review. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6), 865–878.

[219] Prokhorov, D., Zhukov, D., Barinova, O., Anton, K., & Vorontsova, A. (2019). Measuring robustness of visual slam. In *2019 16th International Conference on Machine Vision Applications (MVA)* (pp. 1–6).: IEEE.

[220] Radwan, N., Valada, A., & Burgard, W. (2018). Vlocnet++: Deep multitask learning for semantic visual localization and odometry. *IEEE Robotics and Automation Letters*, 3(4), 4407–4414.

[221] Raguram, R., Frahm, J.-M., & Pollefeys, M. (2008). A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus. In *European Conference on Computer Vision* (pp. 500–513).: Springer.

[222] Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

[223] Ranjan, A., Jampani, V., Balles, L., Kim, K., Sun, D., Wulff, J., & Black, M. J. (2019). Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 12240–12249).

[224] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779–788).

[225] Redmon, J. & Farhadi, A. (2017). Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7263–7271).

[226] Redmon, J. & Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

[227] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91–99).

[228] Revaud, J., Weinzaepfel, P., de Souza, C., & Humenberger, M. (2019). R2D2: Repeatable and reliable detector and descriptor. *Advances in Neural Information Processing Systems*, 32.

[229] Riazuelo, L., Montano, L., & Montiel, J. (2017). Semantic visual slam in populated environments. In *2017 European Conference on Mobile Robots (ECMR)* (pp. 1–7).: IEEE.

[230] Riebesell, J. (2022). tikz. `https://github.com/janosh/tikz`. GitHub repository.

[231] Robbins, H. & Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, (pp. 400–407).

[232] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In N. Navab, J. Hornegger, W. M. W. III, & A. F. Frangi (Eds.), *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, volume 9351 of *Lecture Notes in Computer Science* (pp. 234–241).: Springer.

[233] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6), 386.

[234] Rosten, E. & Drummond, T. (2006). Machine learning for high-speed corner detection. In *European conference on computer vision* (pp. 430–443).: Springer.

[235] Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision* (pp. 2564–2571).

[236] Salas-Moreno, R. F., Newcombe, R. A., Strasdat, H., Kelly, P. H., & Davison, A. J. (2013). Slam++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1352–1359).

[237] Sanchez-Lopez, J. L., Fernández, R. A. S., Bavle, H., Sampedro, C., Molina, M., Pestana, J., & Campoy, P. (2016). Aerostack: An architecture and open-source software

framework for aerial robotics. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)* (pp. 332–341).: IEEE.

[238] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4510–4520).

[239] Sarlin, P.-E., Cadena, C., Siegwart, R., & Dymczyk, M. (2019). From coarse to fine: Robust hierarchical localization at large scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 12716–12725).

[240] Sarlin, P.-E., DeTone, D., Malisiewicz, T., & Rabinovich, A. (2020). Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 4938–4947).

[241] Sattler, T., Leibe, B., & Kobbelt, L. (2011). Fast image-based localization using direct 2d-to-3d matching. In *International Conference on Computer Vision* (pp. 667–674).: IEEE.

[242] Sattler, T., Leibe, B., & Kobbelt, L. (2012). Improving image-based localization by active correspondence search. In *European conference on computer vision* (pp. 752–765).: Springer.

[243] Sattler, T., Leibe, B., & Kobbelt, L. (2017). Efficient & effective prioritized matching for large-scale image-based localization. *IEEE transactions on pattern analysis and machine intelligence*, 39(9), 1744–1756.

[244] Scaramuzza, D. & Fraundorfer, F. (2011). Visual odometry [tutorial]. *IEEE Robotics Autom. Mag.*, 18(4), 80–92.

[245] Scaramuzza, D. & Zhang, Z. (2019). Visual-inertial odometry of aerial robots. *arXiv preprint arXiv:1906.03289*.

[246] Scharstein, D. & Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1), 7–42.

[247] Schonberger, J. L. & Frahm, J.-M. (2016). Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4104–4113).

[248] Schönberger, J. L., Price, T., Sattler, T., Frahm, J.-M., & Pollefeys, M. (2016a). A vote-and-verify strategy for fast spatial verification in image retrieval. In *Asian Conference on Computer Vision (ACCV)*.

[249] Schönberger, J. L., Zheng, E., Pollefeys, M., & Frahm, J.-M. (2016b). Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*.

[250] Shakhatreh, H., Sawalmeh, A. H., Al-Fuqaha, A., Dou, Z., Almaita, E., Khalil, I., Othman, N. S., Khreishah, A., & Guizani, M. (2019). Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges. *Ieee Access*, 7, 48572–48634.

[251] Shen, T., Luo, Z., Zhou, L., Deng, H., Zhang, R., Fang, T., & Quan, L. (2019). Beyond photometric loss for self-supervised ego-motion estimation. In *2019 International Conference on Robotics and Automation (ICRA)* (pp. 6359–6365).: IEEE.

[252] Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., & Fitzgibbon, A. (2013). Scene coordinate regression forests for camera relocalization in rgb-d images. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2930–2937).

[253] Siciliano, B., Khatib, O., & Kröger, T. (2008). *Springer handbook of robotics*, volume 200. Springer.

[254] Sifre, L. & Mallat, S. (2014). Rigid-motion scattering for texture classification. *arXiv preprint arXiv:1403.1687*.

[255] SimonWaldherr (2018). Ryze tello. https://commons.wikimedia.org/wiki/File:Ryze_Tello.jpg. Accessed: 2022-04-20, CC BY-SA 4.0.

[256] Simonyan, K., Vedaldi, A., & Zisserman, A. (2013). Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

[257] Smilkov, D., Thorat, N., Kim, B., Viégas, F., & Wattenberg, M. (2017). Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*.

[258] Smith, S. L., Kindermans, P.-J., Ying, C., & Le, Q. V. (2017). Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*.

[259] Smith, S. L. & Le, Q. V. (2017). A bayesian perspective on generalization and stochastic gradient descent. *arXiv preprint arXiv:1710.06451*.

[260] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1), 1929–1958.

[261] Strasdat, H., Montiel, J., & Davison, A. J. (2010). Scale drift-aware large scale monocular slam. *Robotics: Science and Systems VI*, 2(3), 7.

[262] Strasdat, H., Montiel, J. M. M., & Davison, A. J. (2012). Visual SLAM: why filter? *Image Vis. Comput.*, 30(2), 65–77.

[263] Stühmer, J., Gumhold, S., & Cremers, D. (2010). Real-time dense geometry from a handheld camera. In *Joint Pattern Recognition Symposium* (pp. 11–20).: Springer.

[264] Stutz, D. (2020). latex-resources. https://github.com/davidstutz/latex-resources. GitHub repository.

[265] Sucar, E., Liu, S., Ortiz, J., & Davison, A. J. (2021). imap: Implicit mapping and positioning in real-time. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 6229–6238).

[266] Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (pp. 3319–3328).: JMLR. org.

[267] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[268] Taketomi, T., Uchiyama, H., & Ikeda, S. (2017a). Visual SLAM algorithms: a survey from 2010 to 2016. *IPSJ Trans. Comput. Vis. Appl.*, 9, 16.

[269] Taketomi, T., Uchiyama, H., & Ikeda, S. (2017b). Visual slam algorithms: A survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, 9(1), 16.

[270] Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., & Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[271] Tan, M. & Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*.

[272] Tan, M., Pang, R., & Le, Q. V. (2020). Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10781–10790).

[273] Tang, C. & Tan, P. (2019). BA-Net: Dense bundle adjustment networks. *7th International Conference on Learning Representations, ICLR 2019*.

[274] Tang, J., Ericson, L., Folkesson, J., & Jensfelt, P. (2019). Gcnv2: Efficient correspondence prediction for real-time slam. *IEEE Robotics and Automation Letters*, 4(4), 3505–3512.

[275] Tang, J., Folkesson, J., & Jensfelt, P. (2018). Geometric correspondence network for camera motion estimation. *IEEE Robotics and Automation Letters*, 3(2), 1010–1017.

[276] Tang, Y., Zhao, C., Wang, J., Zhang, C., Sun, Q., Zheng, W., Du, W., Qian, F., & Kurths, J. (2020). An overview of perception and decision-making in autonomous systems in the era of learning. *arXiv preprint arXiv:2001.02319*.

[277] Tateno, K., Tombari, F., Laina, I., & Navab, N. (2017). CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-Janua, 6565–6574.

[278] Teed, Z. & Deng, J. (2018). DeepV2D: Video to Depth with Differentiable Structure from Motion. *arXiv preprint arXiv:1812.04605*, (pp. 1–20).

[279] Teed, Z. & Deng, J. (2020). Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision* (pp. 402–419).: Springer.

[280] Teed, Z. & Deng, J. (2021). DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras. *arXiv preprint arXiv:2108.10869*.

[281] Tesla, Inc. (2021). Transitioning to tesla vision. `https://www.tesla.com/support/transitioning-tesla-vision`. Accessed: 2022-04-19.

[282] Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic robotics*. Intelligent robotics and autonomous agents. MIT Press.

[283] Tiwari, L., Ji, P., Tran, Q.-H., Zhuang, B., Anand, S., & Chandraker, M. (2020). Pseudo rgb-d for self-improving monocular slam and depth prediction. In *European Conference on Computer Vision* (pp. 437–455).: Springer.

[284] Torr, P., Fitzgibbon, A. W., & Zisserman, A. (1998). Maintaining multiple motion model hypotheses over many views to recover matching and structure. In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)* (pp. 485–491).: IEEE.

[285] Tran, N. H. & Nguyen, V.-H. (2021). A tightly-coupled visual-inertial system with synchronized time for indoor localization. *International Journal of Mechanical Engineering and Robotics Research*, 10(4).

[286] Triggs, B., McLauchlan, P. F., Hartley, R. I., & Fitzgibbon, A. W. (1999). Bundle adjustment - A modern synthesis. In B. Triggs, A. Zisserman, & R. Szeliski (Eds.), *Vision Algorithms: Theory and Practice, International Workshop on Vision Algorithms, held during ICCV '99, Corfu, Greece, September 21-22, 1999, Proceedings*, volume 1883 of *Lecture Notes in Computer Science* (pp. 298–372).: Springer.

[287] Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, 104(2), 154–171.

[288] Umeyama, S. (1991). Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(04), 376–380.

[289] Ummenhofer, B., Zhou, H., Uhrig, J., Mayer, N., Ilg, E., Dosovitskiy, A., & Brox, T. (2017). DeMoN: Depth and Motion Network for Learning Monocular Stereo. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 5622–5631).: IEEE.

[290] Valada, A., Radwan, N., & Burgard, W. (2018). Deep auxiliary learning for visual localization and odometry. In *IEEE International Conference on Robotics and Automation* (pp. 6939–6946).: IEEE.

[291] Van de Sande, K. E., Uijlings, J. R., Gevers, T., & Smeulders, A. W. (2011). Segmentation as selective search for object recognition. In *2011 international conference on computer vision* (pp. 1879–1886).: IEEE.

[292] Vijayanarasimhan, S., Ricco, S., Schmid, C., Sukthankar, R., & Fragkiadaki, K. (2017). Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*.

[293] Viola, P. & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1 (pp. I–I).: IEEE.

[294] Vödisch, N., Cattaneo, D., Burgard, W., & Valada, A. (2022). Continual slam: Beyond lifelong simultaneous localization and mapping through continual learning. *arXiv preprint arXiv:2203.01578*.

[295] Vogiatzis, G. & Hernández, C. (2011). Video-based, real-time multi-view stereo. *Image Vis. Comput.*, 29(7), 434–441.

[296] Walch, F., Hazirbas, C., Leal-Taixe, L., Sattler, T., Hilsenbeck, S., & Cremers, D. (2017). Image-based localization using lstms for structured feature correlation. In *Camera relocalization by computing pairwise relative poses using convolutional neural network* (pp. 627–637).

[297] Wang, C., Buenaposada, J. M., Zhu, R., & Lucey, S. (2018). Learning depth from monocular videos using direct methods. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2022–2030).

[298] Wang, C., Wang, Y.-P., & Manocha, D. (2021). Motionhint: Self-supervised monocular visual odometry with motion constraints. *arXiv preprint arXiv:2109.06768*.

[299] Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4), 600–612.

[300] Wang, Z., Zhang, Q., Li, J., Zhang, S., & Liu, J. (2019). A computationally efficient semantic slam solution for dynamic scenes. *Remote Sensing*, 11(11), 1363.

[301] Wangsiripitak, S. & Murray, D. W. (2009). Avoiding moving outliers in visual slam by tracking moving objects. In *2009 IEEE international conference on robotics and automation* (pp. 375–380).: IEEE.

[302] Weyand, T., Kostrikov, I., & Philbin, J. (2016). Planet-photo geolocation with convolutional neural networks. In *European Conference on Computer Vision* (pp. 37–55).: Springer.

[303] Wistuba, M., Rawat, A., & Pedapati, T. (2019). A survey on neural architecture search. *arXiv preprint arXiv:1905.01392*.

[304] Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., & Keutzer, K. (2019). Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 10734–10742).

[305] Wu, C. et al. (2011). Visualsfm: A visual structure from motion system.

[306] Wu, J., Ma, L., & Hu, X. (2017). Delving deeper into convolutional neural networks for camera relocalization. In *IEEE International Conference on Robotics and Automation* (pp. 5644–5651).: IEEE.

[307] Yang, N., von Stumberg, L., Wang, R., & Cremers, D. (2020a). D3VO: deep depth, deep pose and deep uncertainty for monocular visual odometry. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020* (pp. 1278–1289).: Computer Vision Foundation / IEEE.

[308] Yang, N., Wang, R., Stuckler, J., & Cremers, D. (2018). Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 817–833).

[309] Yang, T.-Y., Nguyen, D.-K., Heijnen, H., & Balntas, V. (2020b). UR2KiD: Unifying Retrieval, Keypoint Detection, and Keypoint Description without Local Correspondence Supervision. *arXiv preprint arXiv:2001.07252*.

[310] Yi, K. M., Trulls, E., Lepetit, V., & Fua, P. (2016). Lift: Learned invariant feature transform. In *European conference on computer vision* (pp. 467–483).: Springer.

[311] Yin, Z. & Shi, J. (2018). Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1983–1992).

[312] Zeiler, M. D. & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818–833).: Springer.

[313] Zhan, H., Garg, R., Weerasekera, C. S., Li, K., Agarwal, H., & Reid, I. D. (2018). Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018* (pp. 340–349).: Computer Vision Foundation / IEEE Computer Society.

[314] Zhan, H., Weerasekera, C. S., Bian, J., Garg, R., & Reid, I. D. (2021). DF-VO: what should be learnt for visual odometry? *CoRR*, abs/2103.00933.

[315] Zhang, H., Wang, G., Lei, Z., & Hwang, J.-N. (2019). Eye in the sky: Drone-based object tracking and 3d localization. In *Proceedings of the 27th ACM International Conference on Multimedia* (pp. 899–907).

[316] Zhao, C., Sun, L., Purkait, P., Duckett, T., & Stolkin, R. (2018). Learning monocular visual odometry with dense 3d mapping from dense 3d flow. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 6864–6871).: IEEE.

[317] Zhao, H., Gallo, O., Frosio, I., & Kautz, J. (2015). Is l2 a good loss function for neural networks for image processing? arxiv preprint. *arXiv preprint arXiv:1511.08861*.

[318] Zhao, Q., Sheng, T., Wang, Y., Tang, Z., Chen, Y., Cai, L., & Ling, H. (2019a). M2det: A single-shot object detector based on multi-level feature pyramid network. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33 (pp. 9259–9266).

[319] Zhao, W., Chellappa, R., Phillips, P. J., & Rosenfeld, A. (2003). Face recognition: A literature survey. *ACM computing surveys (CSUR)*, 35(4), 399–458.

[320] Zhao, W., Liu, S., Shu, Y., & Liu, Y. (2020). Towards better generalization: Joint depth-pose learning without posenet. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020* (pp. 9148–9158).: Computer Vision Foundation / IEEE.

[321] Zhao, Z.-Q., Zheng, P., Xu, S.-t., & Wu, X. (2019b). Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11), 3212–3232.

[322] Zhou, H., Ummenhofer, B., & Brox, T. (2018a). DeepTAM: Deep Tracking and Mapping. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11220 LNCS, 851–868.

[323] Zhou, Q.-Y., Park, J., & Koltun, V. (2018b). Open3D: A modern library for 3D data processing. *arXiv:1801.09847*.

[324] Zhou, T., Brown, M., Snavely, N., & Lowe, D. G. (2017). Unsupervised learning of depth and ego-motion from video. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017* (pp. 6612–6619).: IEEE Computer Society.

[325] Zhou, Y., Barnes, C., Lu, J., Yang, J., & Li, H. (2019a). On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 5745–5753).

[326] Zhou, Y., Yan, F., & Zhou, Z. (2019b). Handling pure camera rotation in semi-dense monocular SLAM. *Vis. Comput.*, 35(1), 123–132.

[327] Zhu, Z., Peng, S., Larsson, V., Xu, W., Bao, H., Cui, Z., Oswald, M. R., & Pollefeys, M. (2021). Nice-slam: Neural implicit scalable encoding for slam. *arXiv preprint arXiv:2112.12130*.

[328] Zitnick, C. L. & Dollár, P. (2014). Edge boxes: Locating object proposals from edges. In *European conference on computer vision* (pp. 391–405).: Springer.

[329] Zou, Y., Luo, Z., & Huang, J.-B. (2018). Df-net: Unsupervised joint learning of depth and flow using cross-task consistency. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 36–53).

[330] Zou, Z., Shi, Z., Guo, Y., & Ye, J. (2019). Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*.