Universidade Estadual de Campinas
Instituto de Computação

Iury Cleveston

RAM-VO: A Recurrent Attentional Model for Visual Odometry

RAM-VO: Um Modelo Atencional Recorrente para Odometria Visual

CAMPINAS
2021

Iury Cleveston

RAM-VO: A Recurrent Attentional Model for Visual Odometry

RAM-VO: Um Modelo Atencional Recorrente para Odometria Visual

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Dissertation presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Computer Science.

Supervisor/Orientadora: Profa. Dra. Esther Luna Colombini

Este exemplar corresponde à versão final da Dissertação defendida por Iury Cleveston e orientada pela Profa. Dra. Esther Luna Colombini.

CAMPINAS
2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

C599r

Cleveston, Iury, 1994-
RAM-VO : a recurrent attentional model for visual odometry / Iury Cleveston. – Campinas, SP : [s.n.], 2021.

Orientador: Esther Luna Colombini.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Odometria visual. 2. Aprendizado profundo. 3. Visão por computador. 4. Aprendizado por reforço profundo. 5. Robótica. I. Colombini, Esther Luna, 1980-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: RAM-VO : um modelo atencional recorrente para odometria visual
Palavras-chave em inglês:
Visual odometry
Deep learning
Computer vision
Deep reinforcement learning
Robotics
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Esther Luna Colombini [Orientador]
Eduardo Todt
Marcos Ricardo Omena de Albuquerque Maximo
Data de defesa: 07-05-2021
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-6010-4624
- Currículo Lattes do autor: http://lattes.cnpq.br/8544906785454562

Universidade Estadual de Campinas
Instituto de Computação

Iury Cleveston

RAM-VO: A Recurrent Attentional Model for Visual Odometry

RAM-VO: Um Modelo Atencional Recorrente para Odometria Visual

Banca Examinadora:

- Profa. Dra. Esther Luna Colombini
  Instituto de Computação - Universidade Estadual de Campinas
- Prof. Dr. Eduardo Todt
  Departamento de Informática - Universidade Federal do Paraná
- Prof. Dr. Marcos Ricardo Omena de Albuquerque Máximo
  Divisão de Ciência da Computação - Instituto Tecnológico da Aeronáutica

A ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 07 de maio de 2021

Those who can imagine anything, can create the impossible.

(Alan Turing)

Acknowledgements

First, I praise the Lord for giving us life and knowledge to make our dreams come true.

I thank my advisor, Prof. Esther Luna Colombini, who, from the first day, has been committed to passing on her knowledge and has made this work the best possible. Esther also made this graduation course the most enjoyable and rich I could ever imagine. Thank you for your trust and partnership during this time.

I thank my girlfriend, Alana Santana, for supporting and giving me the strength during the hard times and for the fruitful ideas about this work. I am grateful to my parents Vilmar and Janete Cleveston, and my sister Esther Cleveston, for the love, support, and encouragement in all these days. You have always been the inspiration and motivation for me.

I thank my friends in the Laboratory of Robotics and Cognitive Systems (LaRoCS) for providing me an excellent place for learning and conditions to complete this work. Also, I thank the professors at the University of Campinas (Unicamp) who shared their knowledge and enthusiasm during classes. My sincere acknowledgment to all the people involved in my day-to-day activities during this course and contributed to this work.

Finally, I thank Bradesco Bank for providing the financial conditions to maintain my studies and complete this work. This study was also financed in part by the Brazilian National Council for Scientific and Technological Development (CNPq), grant 130834/2019-0.

Thank you all!

Resumo

Construir veículos capazes de operar sem supervisão humana é um grande desafio, exigindo percepção e compreensão adequadas do mundo. Neste sentido, determinar a posição espacial do veículo é fundamental. Algoritmos de Odometria Visual (VO) estimam a postura do agente usando apenas alterações visuais nas imagens de entrada. Os métodos de odometria visual mais recentes implementam técnicas de deep learning usando redes neurais convolucionais (CNN) extensivamente, o que adiciona um custo substancial ao lidar com imagens de alta resolução. Em tarefas de VO, mais dados de entrada não significa uma melhor previsão; pelo contrário, a rede pode ter que aprender a filtrar informações desnecessárias. Portanto, a implementação de arquiteturas computacionalmente leves e eficientes despertou o interesse em abordar o problema a partir de uma nova perspectiva. Neste contexto, o Modelo Recorrente Atencional (RAM) surge como uma nova arquitetura, que implementa o conceito de atenção através da seleção de partes essenciais da informação usando aprendizado por reforço (RL). No entanto, o RAM foi introduzido principalmente como prova de conceito para tarefas de classificação no conjunto de dados MNIST. Neste trabalho, propomos o RAM-VO, que é a extensão do RAM para tarefas de regressão e odometria visual. A nova arquitetura modifica a arquitetura base e melhora a representação visual e temporal das informações, incluindo o fluxo ótico como informação contextual para inicialização do agente de RL. Além disso, o RAM-VO implementa o algoritmo Proximal Policy Optimization (PPO) no lugar do algoritmo REINFORCE, o que garante o aprendizado de uma política mais robusta. Os resultados indicam que o RAM-VO pode realizar regressões com seis graus de liberdade a partir de imagens de entrada monoculares usando aproximadamente 3 milhões de parâmetros. Além disso, experimentos no conjunto de dados KITTI demonstram que o RAM-VO alcança resultados competitivos utilizando apenas 5.7% da informação visual disponível.

Abstract

Building vehicles capable of operating without human supervision is challenging, requiring a proper perception and understanding of the world. Mainly, determining the vehicle's pose is fundamental. Visual Odometry (VO) algorithms estimate the agent's egomotion using only visual changes from the input images. The most recent visual odometry methods implement deep-learning techniques using convolutional neural networks (CNN) extensively, which add a substantial cost when dealing with high-resolution images. In VO tasks, more input data does not mean a better prediction; on the contrary, the network may have to learn how to filter out useless information. Therefore, the implementation of computationally efficient and lightweight architectures has sparked an interest in approaching the problem from a new perspective. In this context, the Recurrent Attention Model (RAM) has emerged as a novel architecture, which implements the concept of attention by incrementally selecting the essential pieces of information using reinforcement learning. However, RAM was introduced mainly as a concept proof for classification tasks on the MNIST dataset. In this work, we propose the RAM-VO, which is the RAM's extension to regression and visual odometry tasks. Our novel model modifies the basic RAM architecture and improves the visual and temporal representation of information, including the optical flow as contextual information for initializing the RL agent. Also, RAM-VO implements the Proximal Policy Optimization (PPO) algorithm in place of the REINFORCE algorithm, which guarantees the learning of a robust policy. The results indicate that RAM-VO can perform regressions with six degrees of freedom from monocular input images using approximately 3 million parameters. In addition, experiments on the KITTI dataset demonstrate that RAM-VO achieves competitive results using only 5.7% of the available visual information.

List of Figures

List of Tables

List of Abbreviations and Acronyms

ATE	Absolute Trajectory Error
BA	Bundle Adjustment
CAT	Canonical Appearance Transformations
CL	Curriculum Learning
CNN	Convolutional Neural Network
DL	Deep Learning
DoF	Degree of Freedom
DSO	Direct Sparse Odometry
DTAM	Dense Tracking and Mapping
EKF	Extended Kalman Filter
GAN	Generative Adversarial Network
HOG	Histogram of the Oriented Gradients
IMU	Inertial Measurement Unit
KD	Knowledge Distillation
LSTM	Long Short-term Memory
MDP	Markov Decision Process
MSE	Mean Squared Error
PnP	Perspective from n Points
PPO	Proximal Policy Optimization
RAM	Recurrent Attentional Model
RANSAC	Random Sample Consensus
RL	Reinforcement Learning
RPE	Relative Pose Error
VO	Visual Odometry

Contents

Chapter 1

Introduction

Autonomous vehicles have attracted significant attention in the last years. Several companies started commercializing these vehicles, notably autonomous cars and drones, bringing more traction and funding to the field. Further, autonomous vehicles have been a long-standing research topic for the scientific community. Building vehicles capable of operating without human supervision is challenging, which requires a proper perception and understanding of the world. Although many tasks depend on these concepts, determining the vehicle's spatial position is fundamental.

Visual Odometry (VO) is the field concerned with estimating the egomotion of an agent (e.g., human, automobile, drone, robot) using only visual changes from the input images. Nistér et al. [91] named the field in reference to classical odometry, which uses wheel encoders' information. VO emerged with the promise of solving the main issues that wheel odometry presents, such as the pose estimation error due to wheel skating, skidding, and displacement over uneven terrain. The results achieved by visual odometry also provided the ability to estimate the pose in environments where GPS cannot be employed (e.g., aerospace, underwater, indoor); and replace expensive laser scanner sensors. Compared to traditional sensors (e.g., sonars, accelerometers, and IMUs), cameras provide richer information, which can be relevant to the embedded application [107].

Several approaches have been developed to estimate the agent's pose by capturing the apparent motion from subsequent images. These methods require the environment to have sufficient light, the objects to have texture, and the subsequent images to overlap. These methods can be classified as direct or indirect, depending on how they process the input image. Direct methods operate directly on the pixel intensities, using more information from the scene in exchange for processing power and speed. Indirect methods detect and track high-salient features in the frames; they are faster but discard valuable information.

Both methods present advantages and disadvantages. However, hand-crafted solutions based on either direct or indirect approaches become complicated due to the problem's nature, which presents a huge search space, and considerable non-linearities. Usually, these solutions do not cover all possibilities and suffer from robustness issues when used in different environments. Even traditional learning methods such as kernel machines, Gaussian processes, and linear regression could not provide reliable solutions in a real-world scene [17, 47, 100].

In recent years, Deep Learning (DL) techniques have appeared as a novel way to

perform statistical learning on real data captured from the environment. The use of DL has shown promising results in complex datasets, such as KITTI [42], TUM [109], and EuRoC [10]. These datasets impose a great challenge for traditional methods due to sudden changes in the agent's speed, changes in the scene such as illumination, shadows, occlusions, and simultaneous motion of numerous objects. Nevertheless, DL techniques can solve these problems by learning the various nonlinear factors that influence the scene generation and motion, outperforming traditional learning methods that usually suffer from non-linearities [45].

However, the visual odometry field makes extensive use of convolutional neural networks (CNN), which add a substantial cost when dealing with high-resolution images. Further, more input data does not mean a better prediction; on the contrary, the network may have to learn how to filter out useless information. Therefore, the implementation of computationally efficient and lightweight architectures, especially for mobile devices, has sparked an interest in approaching the problem from a new perspective. Though capturing only the necessary information is fundamental, learning where to look requires elaborating several cognitive concepts, such as attention and memory. Attention, especially, has quickly attracted the scientific community's interest due to its ability to provide inexpensive solutions to complex problems.

In this context, the Recurrent Attention Model (RAM) [80] have emerged as a novel architecture, which implements a recurrent attentional glimpse, incorporating the attention concept by incrementally selecting the essential pieces of information. One of the RAM's main advantages is employing reinforcement learning (RL) to guide the glimpse sensor through the image; the RL paradigm allows the model to learn a more robust and efficient policy by trial and error. However, RAM was introduced mainly as a concept proof, only implemented for classification tasks on the MNIST [68] dataset. RAM also uses the REINFORCE algorithm [130] to guide the glimpse sensor, but this algorithm presents convergence problems and slowness to learn in challenging scenarios.

In this work, we propose the RAM-VO, an extension of RAM to visual odometry tasks. Compared to other odometry modalities, visual odometry problems are considerably more complicated as input images have high resolution, and visual information is complex. The 6-degrees-of-freedom (DoF) regression is also challenging – the smallest error in angle prediction results in significant drift errors in the end. Therefore, extending the RAM's potential from classification to regression tasks requires a significant change in the architecture and an increase in the model's representation capacity. The novel architecture implements spatial and temporal structures to operate with complex visual inputs; the optical flow is also employed as contextual information to initialize the RL agent. Enabling RAM-VO to learn robust policies requires replacing the REINFORCE algorithm [130] with the Proximal Policy Optimization (PPO) [111]. To the best of our knowledge, this is the first architecture to perform visual odometry that implements reinforcement learning in part of the pipeline.

1.1 Objectives

This work aims to create a monocular end-to-end visual odometry architecture – RAM-VO – that employs the reinforcement learning paradigm to train an attentional focus over time. The proposed RAM-VO architecture will extend the RAM [80] by introducing spatial, temporal, and contextual elements to enable the 6-DoF pose regression in real-world sequences. The RAM-VO is expected to be more computationally efficient than similar VO methods due to the addition of attentional concepts and the use of Proximal Policy Optimization (PPO) [111] to learn the agent's policy.

In this sense, to achieve the main objective, the following specific objectives are proposed:

- To conduct a literature review on visual odometry methods that use deep learning, reinforcement learning, and attention;
- To explore spatial and temporal elements to extend the RAM [80] architecture from classification to regression tasks;
- To design the new visual odometry architecture using the Proximal Policy Optimization (PPO) [111] algorithm and contextual information to perform 6-DoF regression in real-world sequences;
- To validate the proposed architecture in the KITTI dataset [43] in different trajectories.

Therefore, based on the specified objectives, the following hypotheses are raised:

1. H_1: The RAM [80] architecture can be extended to perform regression tasks;
2. H_2: The extended RAM architecture can be utilized to perform regression tasks in complex visual scenarios;
3. H_3: The Proximal Policy Optimization (PPO) [111] can be employed to generate a robust policy for VO regression tasks;
4. H_4: The optical flow can be used as contextual information to initialize the RL agent and to guide the glimpse sensor in the construction of an efficient latent space for VO regression tasks;
5. H_5: The proposed RAM-VO can achieve state-of-the-art results concerning the drift error with a lesser computational cost.

1.2 Contributions

This work provides the following contributions:

- A literature review of the visual odometry methods that use Deep Learning techniques;

- A novel architecture for visual regression, derived from RAM [80];
- A novel visual odometry architecture that implements innovative concepts derived from biology, such as attention;
- The first visual odometry architecture that implements reinforcement learning as part of the pipeline;
- Several experiments on real-world VO sequences demonstrating the validity and efficiency of RAM-VO.

The proposed architecture and codes are available on the following GitHub repository: `https://github.com/icleveston/RAMVO`.

1.3 Text Structure

This work is structured in six chapters to provide the reader with the necessary fundaments for understanding the proposed architecture and the results achieved.

Chapter 1 introduced the problem we intend to solve and indicated our motivation and objectives with this work. Also, hypotheses were raised to guide our investigation. We additionally presented the contributions that this work gave to the field.

In Chapter 2, we will address the theoretical basis for visual odometry, reinforcement learning, and the Recurrent Attention Model [80], which will serve as the foundation for comprehending this work.

Chapter 3 contains a literature review of the most used visual odometry methods, such as the learning-based methods. Besides, we present a review of the methods that use concepts of attention in their construction.

Chapter 4 contains the materials used to construct our proposed model, such as datasets for training models, metrics for evaluation, and software and hardware technologies. Also, we will describe in detail the methodology used in the development.

Chapter 5 details the RAM-VO development. We start by extending the RAM architecture to regression tasks while adding more sophisticated structures to represent spatial, temporal, and contextual information. We also explore alternative RL methods to learn better policies, and discuss the experiments in visual odometry sequences and compare them with the literature.

Finally, Chapter 6 concludes this work, assessing our results and providing directions for further improvements and possible future work.

Chapter 2

Theoretical Background

This chapter presents the fundamental knowledge to comprehend the proposed architecture and the achieved results. We will explain the visual odometry theory, camera models, and egomotion estimation. We will also introduce the reinforcement learning paradigm and the policy gradient methods such as REINFORCE [130] and Proximal Policy Optimization (PPO) [111] algorithm. Finally, we detail the Recurrent Attention Model [80], which will serve as the base for our work.

2.1 Visual Odometry

Odometry is the process of estimating the agent's egomotion, which is the incremental progression of position and orientation over time. Traditionally, odometry is performed by placing encoders on the robot's wheels and counting the rotations. Wheel odometry is closely related to the robot's kinematic model, extensively employed to determine the pose by computing the incremental displacements from a known starting point. However, the encoder readings regularly present small errors caused mainly by the robot skidding in uneven terrains or slippery floors [136]. These errors accumulate over time, gradually drifting the estimated pose from the ground-truth pose. The specificities of each floor further complicate the development of precise error models.

Incorporating inertial sensors (e.g., inertial measurement units, accelerometers, gyroscopes) can reduce the drift error, although it cannot be removed completely; the error accumulation rate depends on the sensor's precision and quality. GPS and laser scanner sensors can also determine the pose; however, they have limitations regarding the cost, latency, and precision in indoor and outdoor spaces, respectively [107, 11]. Currently, wheel odometry still presents significant accumulating drift error and only works for wheeled ground vehicles, impeding its use for legged robots and aerial vehicles.

With the increase in computational power, an alternative has been to use camera sensors, which became increasingly cheap and accurate in the last few years. Images contain more information than simple wheel encoders readings, enabling real motion detection instead of inferring it from angular velocities. Therefore, visual odometry (VO) methods estimate the egomotion by computing visual changes between the frames [107]. To detect the apparent motion, the environment must present sufficient illumination, texture, and

the captured frames must overlap. VO can only be estimated when there are visual variations across the frames, including individual pixel changes. Also, the camera capturing frequency must be adequately set concerning the agent's dynamics – inappropriate capturing rates might prevent the images from overlapping or generating aliasing, making egomotion estimation inaccurate.

Historically, the VO field emerged around the 1980s with the works of Moravec [84], which captured images from stereo cameras sliding on a rail. Corners were detected in the images and matched using the Normalized Cross-Correlation (NCC) to reconstruct the original 3D structure. Moravec [83] also described one of the first corner detectors, being the predecessor of several popular detectors [39, 51], his corner detector was subsequently applied in VO. Other essential landmarks were reached by Olson et al. [93], which built a maximum-likelihood stereo vision method to reduce the accumulating drift error for long distances in exploratory rovers. Lacroix [67] also worked on rovers able to operate correctly on all kinds of terrains found in planetary-like environments. These rovers should estimate the 6 degrees of freedom (DoF) from tracking features across raw stereo images instead of using geometry constraints for motion estimation. Cheng et al. [15] further applied these ideas to the Mars Rover.

Nistér et al. [91] introduced the term "Visual Odometry" and developed a novel feature-based method to estimate the motion, which detected the features and matched them in all frames – instead of tracking them across the frames. They also proposed an outlier removal method based on the Random Sample Consensus (RANSAC) [37], which eliminates incorrect features during the matching step. Further, Scaramuzza and Siegwart [108] proposed a method to operate directly at the pixel level, which uses a monocular omnidirectional camera for outdoor environments.

In general, a VO solution is usually composed of either a monocular [12] or a stereo camera [77, 3, 90]. A monocular VO uses only one camera, which becomes cheaper and more accessible. Frequently, monocular VO incorporates inertial and dimensional environment information to determine the displacements' absolute scale; an alternative to recover the absolute scale is to use two cameras. Stereo VO explores the camera's epipolar geometry to extract information from two images of the same scene, captured from a slightly different perspective, as shown in Figure 2.1. However, a stereo VO solution requires strict camera synchronization and calibration [107]. Recently, newer VO systems started to include omnidirectional and RGB-D cameras [139, 29, 53].

The formulation of a visual odometry problem considers a rigid-body agent moving in the environment and capturing images in equally sampled time instants k. For simplicity, the coordinate of the camera is assumed to be the same as the agent. Consider the captured image $\mathbf{I}_k = \{\mathbf{I}_k^L, \mathbf{I}_k^R\}$ in a stereo system and $\mathbf{I}_k = \{\mathbf{I}_k^L\}$ in a monocular system. Therefore, the main goal of a VO system is to estimate the relative rigid-body transformation $\mathbf{T}_k \in \mathbb{R}^{4\times4}$ from the images $\mathbf{I}_{k-1}$ and $\mathbf{I}_k$ as

$$\mathbf{T}_k = \begin{bmatrix} \mathbf{R}_k & \mathbf{t}_k \\ 0 & 1 \end{bmatrix}, \tag{2.1}$$

where $\mathbf{R}_k \in \mathbb{R}^{3\times3}$ is the rotation matrix, $\mathbf{t}_k \in \mathbb{R}^{3\times1}$ is the translation vector. The current pose can be determined by multiplying the previous pose by the current transformation

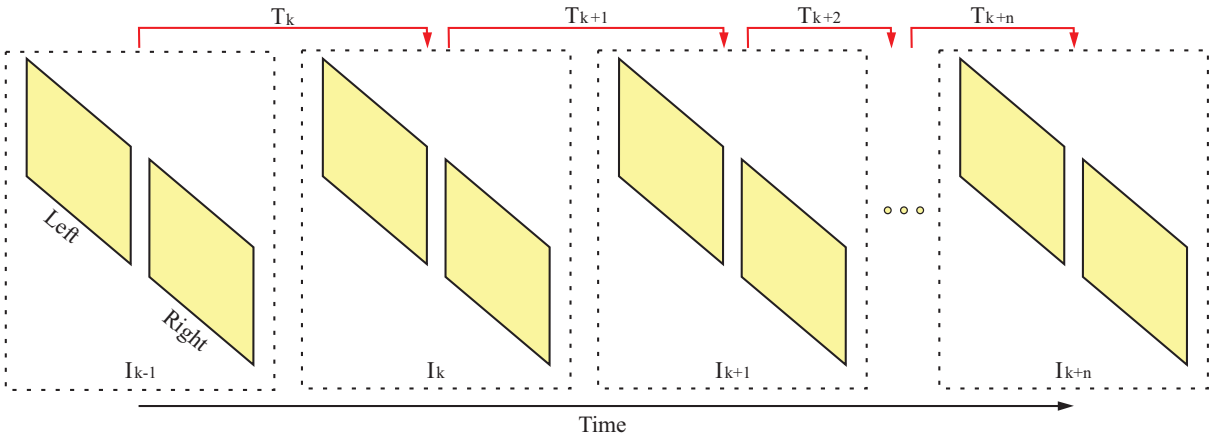


Figure 2.1: An illustration of a stereo VO system, which is composed of a left and right camera. The system captures images at time instants k, and each captured image pair $\{\mathbf{I}_{k-1}, \mathbf{I}_k\}$ is related by a rigid-body transformation $\mathbf{T}_k$.

$\mathbf{T}_k$ as

$$\mathbf{C}_k = \mathbf{C}_{k-1}\mathbf{T}_k, \tag{2.2}$$

with $\mathbf{C}_0$ being the agent's initial pose at instant $k = 0$. The concatenation of all the poses $\mathbf{C}_{0:n} = \{\mathbf{C}_0, \mathbf{C}_1, ..., \mathbf{C}_n\}$ provides the full trajectory. In this sense, a VO system retrieves the path incrementally, pose after pose [107].

The rigid-body transformation $\mathbf{T}_k$ can be estimated by either computing visual variations from the image pixels or the pre-processed features extracted from the image. **Direct** or **appearance-based methods** operate directly on pixel intensities provided by the camera sensor. These methods minimize a photometric error for each frame and are computationally more expensive – most of the DL works for visual odometry implement a direct approach. On the other hand, **indirect** or **feature-based methods** use only salient features extracted from the image, such as corners, edges, and invariant blob features. Indirect methods are fast, but they discard relevant information presented in the pixel's intensities, which becomes problematic in environments with low texture and blur. These methods generally perform a keypoint detection and matching across frames; therefore, the egomotion estimation is essentially geometric. Most of the classical VO methods are indirect, mainly due to the computational cost of real-time pixel-level processing, although newer direct methods are challenging this idea. A common alternative is to create hybrid methods to take advantage of both approaches.

2.1.1 Indirect Approaches

Indirect or feature-based methods extract distinct salient points from the image and describe them using feature vectors. These techniques depend on the image textures and are generally not applicable in environments with low texture (e.g., tunnels, sandy soil, and asphalt) [136]. In general, an indirect VO pipeline is composed of several stages, such as feature detection and description [118], feature tracking [28], and egomotion estimation. The input is either a monocular or a stereo image sequence, whose colorspace can be either

RGB or grayscale, as shown in Figure 2.2.

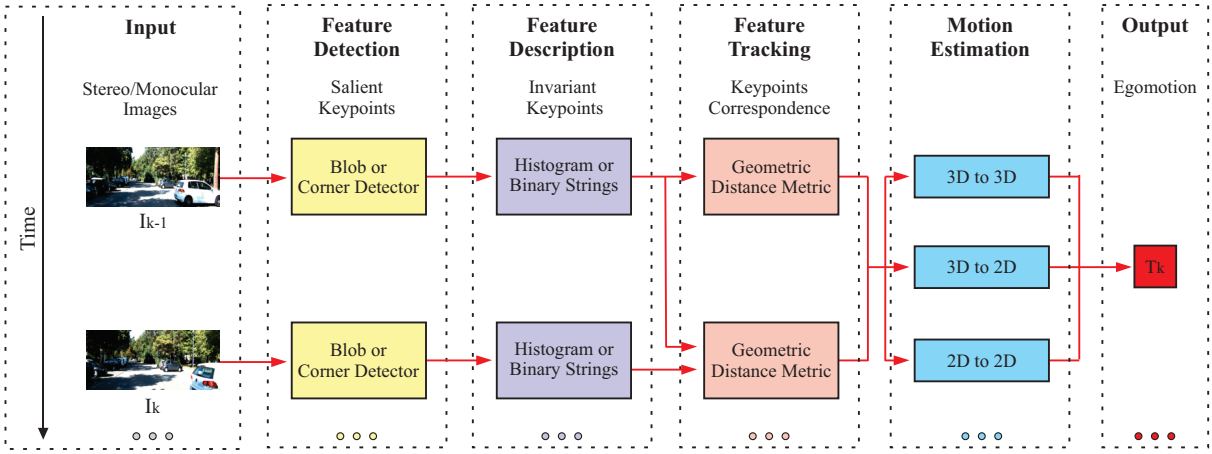


Figure 2.2: An abstraction of a general indirect VO pipeline. The pipeline stages consist of an image sequence as input, feature detection, feature description, feature correspondence and tracking, and egomotion estimation.

The feature detection stage detects the salient keypoints, usually regions of high energy (i.e., where the gradient is high). The detection stage determines the keypoints location using corner detectors [51, 83, 39] or blobs detectors [75, 5, 102]. Most of the blobs detectors also include a descriptor. Traditional edge and corner detection approaches were common for providing fast keypoint correspondence between frames [83, 39, 51]. However, corner detectors are not invariant to several transformations, such as scale, rotation, and translation. The development of invariant blob detectors such as SIFT [75], SURF [5], ORB [102], and BRISK [69] improved the VO accuracy in exchange of speed. These blobs detectors provide much higher robustness to scale and perspective transformations, blur, luminosity, and even small deformations in the images. Consequently, these techniques replaced the simple corner detectors [98, 35, 3]. Despite the wide range of detection methods, selecting a method that correctly meets the desired application latency and precision restrictions is still a challenge. Several works compared the performance of detector-descriptor pairs in visual odometry using different criteria [16, 6, 116, 88]. Pose estimation is significantly influenced by the features selected, which leveraged new research on the feature-selection field, and several works advanced in this direction [23, 25, 61, 9].

The next stage corresponds to the keypoint description, in which the keypoints are described in a dense, useful, and invariant form for future matching across the frames, such as histogram of the oriented gradients (HOG), binary strings, and even raw pixel values. In the feature tracking stage, the descriptors are compared to the subsequent frame descriptors by a geometric distance metric (e.g., Euclidean distance) or correlations. The goal is to find the best feature correspondence between frames; however, the tracking search space can be large and computationally expensive depending on the number of features detected. IMUs, wheel encoder can be employed to reduce the search space; also, data structures (e.g., hash tables, trees) can be utilized to support the tracking.

A necessary procedure is to remove outlier keypoints produced mainly by noise and image deformations. RANSAC [37] is the default method to estimate keypoint correspondence in the presence of outliers. The method works by computing hypotheses from data-

points and discarding the ones that present the lowest consensus with the data. RANSAC returns the model's parameters which best represent the relations between the keypoints. The final stage estimates the egomotion by computing the rigid-body transformation $\mathbf{T}_k$ between consecutive frames using a defined feature's correspondence specification, which can be 3D-to-3D, 3D-to-2D, and 2D-to-2D [40].

Despite the extensive use, indirect methods are still not robust enough for low-texture environments due to limitations in keypoint detection, correspondence robustness, and consistency in providing invariant features. The matches are inadequate in situations where the frame presents blur, brightness variation, or even sudden variations in the agent's speed – resulting in significant estimation errors.

2.1.2 Direct Approaches

Direct or appearance-based methods estimate the rigid-body transformation $\mathbf{T}_k$ by operating directly on the pixel level and computing the variations from the photometric values provided by the camera sensor – instead of using geometric distances between keypoints. Direct methods generally outperform feature-based methods when the scene has low texture due to motion blur or low visibility. In general, a direct VO pipeline is composed of a stage that determines the depth map from the frame $\mathbf{I}_{k-1}$. The epipolar geometry usually calculates this in a stereo setup. Alternatively, several consecutive monocular images can be used in a monocular setup; or a generative model can be employed. Further, the frames $\mathbf{I}_{k-1}$ and $\mathbf{I}_k$ are aligned by iteratively minimizing a nonlinear cost function, as shown in Figure 2.3. It is important to highlight that in learning-based methods, this pipeline is implicit and derived from the data.

Direct methods consider the pixel-intensity gradient magnitude and orientation to minimize the photometric error when aligning the frames $\mathbf{I}_{k-1}$ and $\mathbf{I}_k$. The cost function

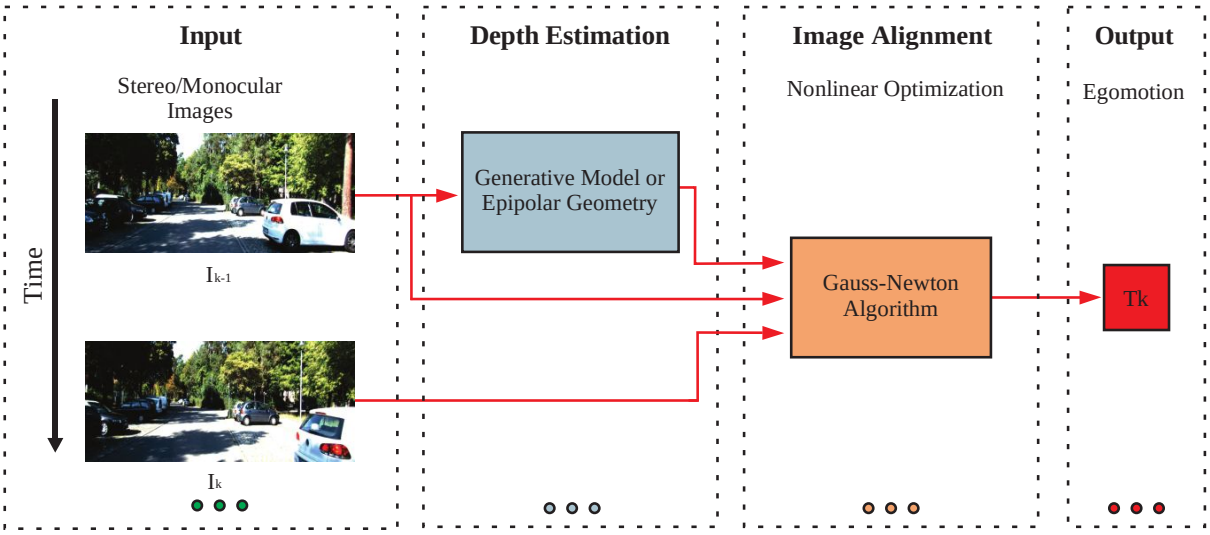


Figure 2.3: An abstraction of a general direct VO pipeline. The pipeline stages consist of a depth map generation for the frame $\mathbf{I}_{k-1}$; an image alignment stage, which optimizes a nonlinear least-squares cost function. The estimated egomotion is the transformation $\mathbf{T}_k$ that minimizes the photometric error during the image alignment stage.

generates a nonlinear least-squares problem, which is frequently solved by the Gauss-Newton algorithm. This method works by iteratively optimizing the function squared sum, giving an initial $\mathbf{T}_k$ guess. Lie algebra is commonly used to reduce the $\mathbf{T}_k$ representation and increase stability and accuracy during optimization [114]. The Gauss-Newton variation called the Levenberg-Marquardt algorithm is frequently used to add the possibility to perform gradient descent [106, 30].

Although the image alignment stage provides reasonable estimates for $\mathbf{T}_k$, the drift error still tends to accumulate over time. One way to mitigate this issue is to refine the egomotion estimation using a technique known as Local Bundle Adjustment (BA). BA is a non-linear simultaneous refinement of structure ($\mathbf{p}$, where $\mathbf{p}$ is a point in the world) and motion, performed by minimizing the cost function considering the last $\mathbf{I}_{k-n}$ frames. Also, direct methods have instability issues during initialization, which requires the generation of a precise depth map. They are also not robust to occlusions in the scene and more computationally expensive when applied to dense image representations. Several approaches try to work around these issues by providing prior knowledge to the cost function, such as smoothing the depth map, geometric constraints for the pixels' location, and providing a sparse representation by segmenting the image in patches.

Historically, the first direct methods only tracked planar image patches, which were manually selected from the scene [38]. Further, a Extended Kalman Filter (EKF) was applied to track the egomotion across the frames [58, 82]. Nonlinear least-squares methods were also utilized to estimate the egomotion [112, 78, 99]. These methods were based on the idea of estimating the patches' normal, which allowed them to track each patch across multiple frames – this technique significantly reduces the drift error compared to indirect methods. Also, these methods provide real-time performance. However, they were limited to small datasets and required extensive image patch selection.

Comport [21] proposed to estimate motion by relaxing the planarity assumption in a stereo setup. Later, the same principles were applied to RGB-D camera sensors [60, 79]. The Dense Tracking and Mapping in Real-Time (DTAM) algorithm [89] introduced a new way to compute the transformation $\mathbf{T}_k$ by using a depth map for each monocular frame. However, DTAM required intensive processing in GPUs due to dense image representation. The work of Engel et al. [31] diminished DTAM's intensive processing cost by using only high-gradient regions. The Direct Sparse Odometry (DSO) [30] proposed a sparse method to compute VO, which combines the photometric error minimization with a consistent joint optimization of all parameters.

2.1.3 The Perspective Camera Model

Visual odometry can be performed with either a perspective or an omnidirectional camera. The classical model for perspective cameras assumes a pinhole projection model, in which the image is formed by the light intersection between the camera's lens center $\mathbf{c}$ and the image plane, as shown in Figure 2.4.

Consider $\mathbf{p} = [x, y, z]^T$ as the 3D point in the world and $\mathbf{q} = [u, v]^T$ as the 2D point projected on the image plane. The mapping of the 3D world to the 2D image is given by

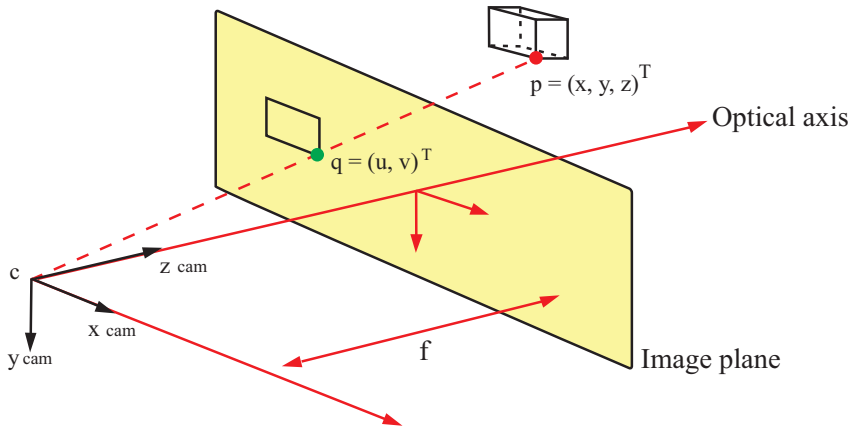


Figure 2.4: Illustration of a perspective camera model. **c** is the camera center; **p** is the 3D world point; **q** is the projected 2D point on the image plane; f is the focal length.

the equation of the perspective projection

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \tag{2.3}$$

where λ is the depth factor and $\mathbf{K}$ is the intrinsic parameters matrix, given by

$$\mathbf{K} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.4}$$

where α_u and α_v are the focal length, u_0, and v_0 are the coordinates of the image plane's center. The intrinsic parameters $\mathbf{K}$ depend on the camera characteristics and do not change from scene to scene. A complete camera modeling can be found in [52].

For a stereo system, the cameras' position and orientation must be determined and precisely calibrated. These parameters are extrinsic, and the most popular calibration method uses a patterned image of a chessboard, in which the position of each square is known. Thus, several chessboard images are captured in different orientations and positions, ensuring the entire field of view is covered. After that, the extrinsic parameters are determined using a minimization method, such as the least-squares method [107].

2.1.4 Egomotion Estimation

The egomotion estimation depends on the type of method employed. Indirect methods rely on the features' correspondence specification. Regularly, the rigid-body transformation $\mathbf{T}_k$ can be determined by three different approaches, such as 3D-to-3D, 3D-to-2D, and 2D-to-2D.

3D-to-3D Indirect Approach. The egomotion is estimated by triangulating the 3D feature points provided by the frame sequence in a stereo system. Further, the Euclidean distance between the points is minimized to find the transformation $\mathbf{T}_k$ between the

frames as

$$\underset{T_k}{\operatorname{argmin}} \sum_i \| \mathbf{X}_k^i - \mathbf{T}_k \mathbf{X}_{k-1}^i \|, \tag{2.5}$$

where $\mathbf{X}_{k-1}$ are the 3D feature points detected in frame $\mathbf{I}_{k-1}$ and $\mathbf{X}_k$ are the 3D features points in frame $\mathbf{I}_k$, i is the number of features used to compute the transformation. Usually, adding more keypoints increases the computational time and presents better robustness.

3D-to-2D Indirect Approach. This approach is similar to 3D-to-3D. However, Nistér [90] defined this approach as more accurate because it minimizes the 2D reprojection error to determine the transformation $\mathbf{T}_k$ instead of the feature position error. The equation is defined by

$$\underset{T_k}{\operatorname{argmin}} \sum_i \| \mathbf{z}^i - f(\mathbf{T}_k, \mathbf{X}_{k-1}^i) \|, \tag{2.6}$$

where $\mathbf{z}$ is the observed feature point in frame $\mathbf{I}_k$, and $f(\mathbf{T}_k, \mathbf{X}_{k-1})$ is the reprojection function, which reprojects the feature point from the 3D world space to the 2D image space. This problem has many solutions, known as "perspective from n points" (PnP) [107]. The egomotion estimation for monocular systems should capture at least three consecutive frames because a triangulation among the points is needed to determine the correspondence.

2D-to-2D Indirect Approach. When 3D points are unavailable, the egomotion can be estimated by exploring the geometry between the cameras, as illustrated by Figure 2.5.

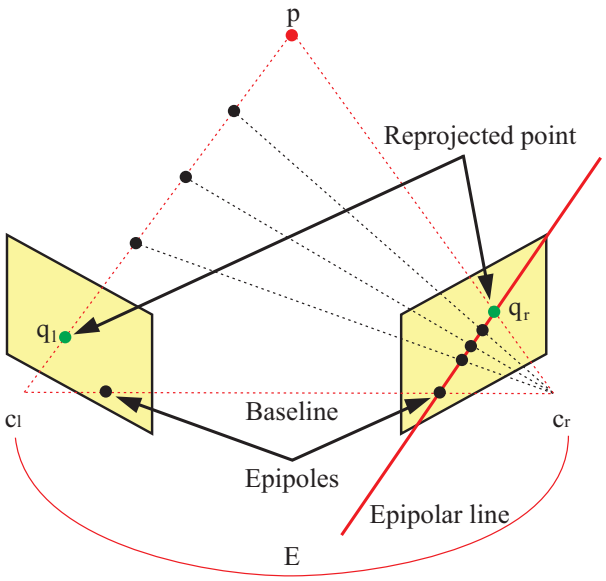


Figure 2.5: An illustration of the epipolar geometry between the left camera $\mathbf{c_l}$ and the right camera $\mathbf{c_r}$. The cameras are related by the essential matrix $\mathbf{E}$ and capture the same 3D point $\mathbf{p}$ at different angles. Adapted from [136].

A stereo system captures the same 3D point $\mathbf{p}$ from the world, at slightly different angles, generating a projection $\mathbf{q_l}$ and $\mathbf{q_r}$ onto both cameras planes centered at $\mathbf{c_l}$ and $\mathbf{c_r}$,

respectively. The cameras are related by a transformation matrix $\mathbf{E}$, which is called the essential matrix, defined by

$$\mathbf{E} = [\mathbf{t}]_{\times}\mathbf{R}, \tag{2.7}$$

where $\mathbf{t} = [t_x, t_y, t_z]^T$ is the translation vector and $\mathbf{R}$ is the rotation matrix. $[\mathbf{t}]\times$ is the skew symmetric matrix defined as

$$[\mathbf{t}]_{\times} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}. \tag{2.8}$$

Therefore, the epipolar geometry can be defined by

$$\mathbf{q_l}\mathbf{E}\mathbf{q_r} = 0. \tag{2.9}$$

The egomotion estimation using epipolar geometry defines a constraint called the epipolar line, in which the points on one image correspond to the points on the other image. Estimating the egomotion for the 2D-to-2D approach requires at least five correspondences of points. Nister [90] proposed the 5-point algorithm to estimate motion. Another alternative is to use the 8-point algorithm [74] or the normalized 8-point algorithm.

Direct Approach. Direct methods usually estimate the rigid-body transformation $\mathbf{T}_k$ by minimizing the photometric error when aligning the previous frame $\mathbf{I}_{k-1}$ with the current frame $\mathbf{I}_k$. Therefore, the log-likelihood or energy ρ are integrated for the entire frame $\mathbf{I}$ concerning each pixel $\mathbf{w} = [u, v]^T$ as

$$\underset{T_k}{\operatorname{argmin}} \int\int_{\mathbf{I}} \rho[\delta I(\mathbf{T}_k, \mathbf{w})] d\mathbf{w}, \tag{2.10}$$

where δI is the difference in pixel intensity defined by the photometric values between the current and previous frame, defined as

$$\delta I(\mathbf{T}_k, \mathbf{w}) = \mathbf{I}_k(\mathbf{K}\mathbf{T}_k\mathbf{K}^{-1}\mathbf{D}(\mathbf{w})\mathbf{w}) - \mathbf{I}_{k-1}(\mathbf{w}). \tag{2.11}$$

The inverse depth map $\mathbf{D}$ should be provided for the time step $k-1$, and the 2D point $\mathbf{w}$ must be visible in both frames.

The process of determining the new pixel coordinate for the frame $\mathbf{I}_k$ can be understood as back-projecting the 2D point $\mathbf{w}$ to the 3D world using the inverse depth map $\mathbf{D}$ and the inverse of intrinsic parameters $\mathbf{K}^{-1}$. Then, the transformation $\mathbf{T}_k$ is applied to the 3D point, rotating and translating it into a new position; after that, the 3D point is projected again to the image plane using the intrinsic parameters $\mathbf{K}$, giving the new 2D coordinate representation.

A common approach is to discretize the image in patches according to the depth map specification [38]. Due to occlusions, a robust cost function δI should be determined. In practice, the cost function is considered as normally distributed, and the problem can be

faced as a least-squares problem, defined by

$$\underset{T_k}{\operatorname{argmin}} \frac{1}{2} \sum_{i=0}^{M} \| \delta I(\mathbf{T}_k, \mathbf{w}_i) \|^2, \tag{2.12}$$

with M corresponding to the set of all image patches previously determined [38].

Equation (2.12) is nonlinear in $\mathbf{T}_k$, which can be commonly solved by the Gauss-Newton method [38]. Also, during the minimization process, the Lie algebra provides a minimal representation for the transformation and increases each iteration's robustness. The $\mathbf{T}_k$ can be mapped into the Lie space by the exponential function as

$$\mathbf{T}(\xi) = \exp(\hat{\xi}), \tag{2.13}$$

where the ξ is called the twist coordinates, represented by $\xi = (\omega, \theta)^T$, where ω is the angular velocity and θ is the linear velocity in the Lie space.

2.2 Reinforcement Learning

Reinforcement Learning (RL) is a machine learning paradigm that consists of an agent that interacts with the environment by taking actions and maximizing a cumulative reward defined by the designer. RL is based on human behavior and optimal control theory but can be quickly extended to several fields (e.g., economics, game theory, information theory). RL is not only limited to classification or regression tasks, but it is also a framework for decision making, knowledge representation, planning, and responding to new and unfamiliar elements [117]. The RL paradigm is fundamental when there is no labeled data available or when the system's dynamics are not differentiable; in such contexts, the model can still learn through rewards signals.

The RL framework consists of several components such as the environment, actions, and rewards, as shown in Figure 2.6. At every time step t, the agent follows a policy $\pi(a|s)$ and performs an action a_t in the environment, receiving a reward r_t and a new state s_{t+1}. The policy is one of the most important aspects of reinforcement learning, which defines the action a that will be taken in each state s. The learning process consists of determining

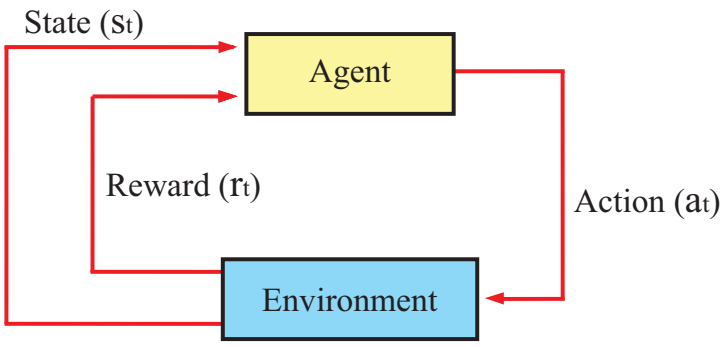


Figure 2.6: Reinforcement learning framework. At every time step t, the agent follows a policy $\pi(a|s)$ and performs an action a_t in the environment, receiving a reward r_t and a new state s_{t+1}.

the optimal policy; in this process, the agent's goal is to maximize the cumulative reward function.

2.2.1 Markov Decision Process (MDP)

Markov Decision Process (MDP) is a discrete-time stochastic control process commonly used to model the environment in reinforcement learning. MDP provides a framework for decision making in which the conditional probability distribution of future states depends only upon the current state; the sequence of previous states does not add new information – this is know as the Markov property. The MDP definition consists of a tuple $M = (S, A, P, R)$, where S is a discrete and finite set of states that model the environment, A is a finite set of actions, $P\left(s^{'}|s, a\right)$ is a probabilistic transition function that describes the effects of executing an action $a \in A$ in a state $s \in S$ and resulting in a state $s' \in S$. $R\left(s, a\right)$ is the reward received by executing an action $a \in A$ in a state $s \in S$. Solving the MDP requires maximizing the total expected reward G_t. Usually, a discount factor γ is used to prevent unbounded accumulated values in non-episodic MDPs, with $0 \leq \gamma < 1$. Hence, after t steps the reward is discounted by γ^t. Therefore, the total expected discounted return is

$$G_t = \sum_{k=0}^{\infty} \gamma^t R_{t+k+1}. \tag{2.14}$$

The solution for an MDP is a policy $\pi : S \rightarrow A$ that specifies the action $a = \pi(s)$ to be chosen in each state s. A policy can be either stationary or non-stationary regarding its variation over time. It can also be deterministic or stochastic, considering the state-action tuple relation. In stationary policies, the best action to be taken in the state s is always the same, regardless of time. In non-stationary policies, the action depends on the step's information. In deterministic policies, each state $s \in S$ is always mapped into a single action; in stochastic policies, states are mapped into a probability distribution of actions; therefore, each action has a probability of being chosen. Among all policies capable of solving an MDP, we desire to find an optimal policy π_* that maximizes the expected total return; this can be done through Value Iteration or Policy Iteration algorithms.

Policy Iteration. Policy iteration-based algorithms find the optimal policy π_* by evaluating and improving a random initial policy π_0 iteratively until convergence. In this class of methods, the policy is evaluated several times in order to approximate the state-value function v_π by

$$v_{k+1}(s) = \sum_{a} \pi(a|s) \sum_{s',r} P\left(s'|s, a\right) \Big[r + \gamma v_k\left(s'\right)\Big]. \tag{2.15}$$

After the state value is updated, the policy is also updated by

$$\pi_{k+1}(s) = \operatorname{argmax} \sum_{s',r} P\left(s'|s, a\right) \Big[r + \gamma v_k\left(s'\right)\Big]. \tag{2.16}$$

Value Iteration. Value iteration-based algorithms determine the state value v_* for each state $s \in S$. They can be seen as an improvement over the Policy Iteration since the

state-value function does not need to be improved only through the policy's improvement. Thus, from an arbitrary v_0, this approach performs updates in all states of S as follows

$$v_{k+1}(s) = \max_{a \in A} \sum_{s',r} P\left(s'|s,a\right) \Big[r + \gamma v_k\left(s'\right) \Big]. \tag{2.17}$$

In the infinite, the state-value function v_k converges to v_*, that is $\lim_{t\to\infty} |v_k\left(s\right) - v_{k-1}\left(s\right)| = 0$; and the optimal policy π_* can be obtained directly from v_*.

POMDP. Most real world problems are not completely accessible to the agent. Hence, they cannot be described as MDPs but rather by Partially Observable Markov Decision Processes (POMDPs), formally defined as a 7-tuple $(S, A, T, R, \Omega, O, \gamma)$, where Ω is the set of observations, O is the set of conditional observation probabilities, and γ is the discount factor. In a POMDP, the agent does not directly observe the environment's true state, but through observations, it updates the probability distribution of the environment is in a specific state s. In an MDP, the agent always knows with confidence the environment's current state, but in a POMDP, the agent also has to learn the belief $b(s)$ after taking action a and observing $o \in \Omega$.

2.2.2 RL Approaches

When the environment's dynamics are unknown or hard to compute, dynamic programming methods based on policy iteration or value iteration become unfeasible. In these cases, different approaches are required. **Monte Carlo** (MC) and **Temporal Difference** (TD) are those approaches.

MC methods require only sequences of samples from states, actions, and rewards of either online or simulated interaction with the environment. This class of methods seeks to solve the learning problem based on the sample returns average. MC methods calculate from complete episodes $S_1, A_1, R_1, ..., S_T$ the return G_t. The main issue with MC methods is that they require an episodic MDP.

On the other hand, TD methods update the estimated value function $V(S_t)$ for state S_t after n-steps by visiting and storing the next n-steps before the update. For the specific case of TD(0), the update is performed immediately after a visit to S_{t+1} through

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)], \tag{2.18}$$

where R_{t+1} is the reward for the next state, α is the learning rate, and $R_{t+1} + \gamma V(S_{t+1})$ is the target for this update. SARSA and Q-learning are two representatives of TD methods.

2.2.3 Policy Gradient Methods

Instead of using value functions to determine the actions, policy gradient methods improve the policy directly by learning a function approximator parameterized by the weights $\boldsymbol{\theta}$. The policy is defined as

$$\pi(a|s, \boldsymbol{\theta}) = Pr\{A_t = a | S_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta}\}, \tag{2.19}$$

which means the probability of action a be taken at time t considering the environment is in state s at time t with parameters $\boldsymbol{\theta}$. The optimization process uses the gradient ascent to update the weights and maximize performance through

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \nabla J(\boldsymbol{\theta}_t), \tag{2.20}$$

where $\nabla J(\boldsymbol{\theta}_t)$ is the stochastic estimate whose expectation approximate the gradient in respect to $\boldsymbol{\theta}$, α is the learning rate, which determine the update magnitude. $J(\boldsymbol{\theta})$ is commonly defined using the value function for the initial state as $v_{\pi_\theta}(s_0)$. Then, the policy gradient theorem states that

$$\nabla J(\boldsymbol{\theta}) = \sum_s d_\pi(s) \sum_a q_\pi(s,a) \nabla_\theta \pi(a|s,\boldsymbol{\theta}), \tag{2.21}$$

where $d_{\pi_\theta(s)}$ is stationary distribution of the Markov chain using π_θ.

The methods that follow this rule for updating the weights are called policy gradient, regardless of whether they used a value function. The configuration of the weights can be made in any way since the policy is differentiable regarding its parameters. An essential point of these methods is their need for exploration; for this reason, the policy cannot become deterministic during training. Commonly, the parameters $\boldsymbol{\theta}$ are represented by the weights between neurons in an artificial neural network. Besides, the policy gradient theorem ensures convergence for this class of methods compared to value-based methods with non-linear function approximators.

The REINFORCE algorithm. The REINFORCE algorithm is directly derived from the policy gradient theorem. The algorithm updates the weights $\boldsymbol{\theta}_t$ proportional to the return G_t in the direction that increases the probability of selecting action A_t in the state S_t, weighted by the action probability. The update rule is given by:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha G_t \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)}, \tag{2.22}$$

where G_t is the cumulative discounted reward at time t.

REINFORCE with Baseline. Being a Monte Carlo method, REINFORCE has high variance, slowing down learning. One way to mitigate this problem is to generalize REINFORCE to compare the return G_t with a baseline function $b(S_t)$. Therefore, the update rule for the REINFORCE with baseline $b(S_t)$ is

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha (G_t - b(S_t)) \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)}. \tag{2.23}$$

The baseline function can be any, as long as it does not vary with the action a; thus, the baseline does not affect the expected value for the update but considerably reduces the variance. For MDPs, the baseline value should depend on the state – generally, the baseline is the estimated value function $\hat{v}(S_t, \mathbf{w})$, where the parameters $\mathbf{w}$ are jointly learned with the policy's parameters $\boldsymbol{\theta}$.

Actor-critic methods. Reinforcement learning methods can also be divided into two classes: actor-only and critic-only. Actor-only methods provide a policy parameterized by gradient ascent. The gradient's performance is directly estimated by simulation, and the

parameters are updated in the direction of improvement; however, these methods present high variance between episodes. Moreover, as the policy changes, a new gradient is estimated regardless of previous estimates; consequently, the accumulation and consolidation of older information do not occur. On the other hand, the critic-only methods depend exclusively on the value function approximation and aim to learn an approximate solution to the Bellman equation ($Q(s_t, a_t) = Q(s_t, a_t) + \alpha(r_t + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t))$), which hopefully will derive an almost optimal policy. These methods do not attempt to optimize the policy space directly.

On the other hand, the actor-critic methods try to combine the strengths of these two methods. The critic learns the value function parameters $\mathbf{w}$, which is then used to update the actor policy's parameters. Depending on the algorithm, the function can be either action/state $q(a|s; \mathbf{w})$ or state/value $v(s; \mathbf{w})$. The actor then updates the policy parameters $\boldsymbol{\theta}$ in the direction suggested by the critic. This strategy maintains the promise of providing faster convergence due to reduced variance, in contrast to critic-only methods for which convergence is guaranteed in limited environments. Summarily, the actor-critic performs the following main steps, as shown by the Algorithm 1.

Algorithm 1: Actor-critic algorithm

Initialize s, $\boldsymbol{\theta}$, $\mathbf{w}$ at random;
Sample $a \sim \pi(a|s; \boldsymbol{\theta})$;
for $t = 1, ..., T$ **do**
  Sample reward $r_t \sim R(s, a)$ and next state $s' \sim P(s'|s, a)$;
  Sample the next action $a' \sim \pi(a'|s'; \boldsymbol{\theta})$;
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_\theta Q(s, a;) \nabla_\theta ln\pi(a|s; \boldsymbol{\theta})$;
  Update policy parameters
  Compute the correction for action-value at time t:
  $G_{t:t+1} = r_t + \gamma Q(s', a'; \boldsymbol{w}) - Q(s, a; \boldsymbol{w})$;
  Use $G_{t:t+1}$ for update value function parameters:
  $\mathbf{w} \leftarrow \mathbf{w} + \alpha_w G_{t:t+1} \nabla_w Q(s, a; \mathbf{w})$;
  Update $a \leftarrow a'$ and $s \leftarrow s'$;
end

2.2.4 The Proximal Policy Optimization (PPO)

The Proximal Policy Optimization (PPO) [111] is an improved version of the TRPO [110] algorithm, which aims to minimize a surrogate function to restrain the size of the policy update's step. TRPO uses the Kullback–Leibler (KL) divergence to determine the variation between the current and old policy; however, PPO simplifies this process by computing the probability ratio and monotonically improves the policy. In essence, the goal is to update the policy parameters inside trusted regions; for that, the objective function is defined as

$$J(\boldsymbol{\theta}) = \mathbb{E}_t\Big[\frac{\pi_{\boldsymbol{\theta}}(a|s)}{\pi_{\boldsymbol{\theta}_{old}}(a|s)} A_t\Big], \tag{2.24}$$

where A_t is the advantage function and $\boldsymbol{\theta}_{old}$ are the old policy's parameters. The use of probability ratios is known as importance sampling, allowing unbiased estimates for the policy's samples. However, importance sampling is unbounded and often causes overestimation and underestimation. One way to solve this is to use the surrogate function, defined as

$$J(\boldsymbol{\theta}) = \mathbb{E}_t \left[\min \left(r(\boldsymbol{\theta}), \text{clip} \left(r(\boldsymbol{\theta}), 1 - \epsilon, 1 + \epsilon \right) \right) A_t \right], \tag{2.25}$$

assuming the probability ratio $r(\boldsymbol{\theta}) = \frac{\pi_{\boldsymbol{\theta}}(a_t, s_t)}{\pi_{\boldsymbol{\theta}_{old}}(a_t, s_t)}$, and ϵ as a hyperparameter commonly defined as 0.2. With this surrogate function, only the overestimation problem is corrected. The underestimation is considered harmless and favors the objective function's concavity.

2.3 Attention

Attention is a concept studied in several fields (e.g., philosophy, psychology, neuroscience, and computer science). Although several studies attempted to define attention, investigating how it occurs and the relation among many cognitive elements is still challenging. One of the most accepted definitions considers attention as a behavioral and cognitive process of focusing selectively on a discrete aspect of information while ignoring other perceptible information [20]. Attention plays an essential role in animal and human cognition. In animals, attention provides the allocation of perceptual resources, which allows them to respond correctly to the environment's stimuli, ensuring their survival. In humans, attention acts on practically all mental processes (e.g., planning, reasoning, and executing).

In this sense, artificial attentional systems based on psychological and neurobiological evidence have existed for at least two decades, intending to provide machines with behaviors closer to humans. Treisman's Feature Integration Theory (FIT) [122], Norman Attentional Model [92, 59], SeLective Attention Model [97], and a Saliency-based Visual Attention [57] are some of the most relevant computational models from an ever-growing list of pioneering research in the field. In the last few years, significant advances in artificial intelligence were obtained by coupling attentional mechanisms in deep neural networks – which is considered one of the most relevant ideas in the field. Such mechanisms were initially used for machine translation and later implemented in computer vision, recommendation systems, question and answer (QA), machine comprehension, automatic summarization, entity recognition, and robotics [121].

Bahdanau et al. [4] proposed the RNNSearch – the first attentional architecture in literature - to solve the information bottleneck problem in encoder-decoder frameworks. The encoder generates a fixed-size context vector in the classic framework, and the decoder used it to generate translations. However, the network needs to compress all the information into a single context vector, deteriorating the learning as the sequence length increases [26]. Therefore, RNNSearch proposes an attentional mechanism to build a dynamic context vector based on the previously translated words and the encoder's hidden states. The attentional mechanism allows all the information from the encoder to be propagated through the network, eliminating the context vector's information bottle-

neck. Based on this approach, several end-to-end attentional networks have appeared in the literature to deal with multiples issues (e.g., sensory multimodality, inter-alignment, long-term dependencies, features recalibration, external memory, and computation time).

Concepts of attention are fundamental to computer vision because this field commonly deals with many nonlinear data. One of the most researched concepts is the multi-glimpse, which refers to iteratively scanning the entire image and finding the most critical regions, enabling the scene understanding with high robustness. Humans build a sequential representation of the analyzed scene by paying attention to only one part of the image simultaneously, enabling a unique performance in capturing the internal relationship of different regions. Also, ignoring irrelevant parts of the image makes the learning process easier in the presence of occlusion, changes in viewpoint, and noise.

However, convolutional neural networks have a very different structure: a) they are not fully scalable and flexible given that the number of parameters increases according to the input size, reaching a point where training may be unfeasible; b) they treat the whole image in the same way for every filtering process, even if the relevant information is only in a small portion of the image; c) they treat equally all feature maps' frequencies, although some may represent only noise or distractors for the task; d) they are massively parallel, neglecting the process of sequential construction of knowledge carried out by human visual attention.

Methods that implement attention in the classic CNN structure aim at improving their structural deficiencies to deal with the above mentioned issues. Attentional mechanisms are introduced mainly in the CNN input and between the convolutional layers. In the input, attention filters the image's raw data, delimiting the most relevant regions for learning; only the selected regions pass to the convolutional layers. Between layers, attention generally acts as a complementary agent to the convolutional operation's deficiencies in capturing dependencies from long distances, internalizing and correctly taking advantage of information from the past, and merging high and low-level features. Liu et al. [72] used attention in CNN to explore important priors in counting crowded tasks by providing attentive maps referring to the crowd regions and each region's density – such information provides much more accurate estimation than traditional approaches. Also, Squeeze-and-Excitation Networks [56] is the first approach to modeling interdependencies between channels to recalibrate features in a mechanism called squeeze and excitation, which is capable of exploiting local dependencies on the channel and capturing channel-wise dependencies in a not mutually exclusive way between channels, introducing in the literature an attentional dynamics to boost features.

Despite the success, few methods explore the composition of glimpse mechanisms, visual attention, and recurrent processing. Fu et al. [41] propose a pioneering framework for recurrent convolutional networks, in which an input image is given, and the classic CNN extracts the feature maps while the attentional structure estimates the next focus region to be served to the CNN instance. Once the focus region is determined, the system crops and enlarges the region to a higher resolution to extract more refined features. In this ensembled structure, each CNN in the stack generates a prediction so that the deeper CNNs generate more accurate predictions. This framework presents a straightforward glimpse mechanism in which attention facilitates the ensemble process. However, the

structure does not fully incorporate the biological mechanism since the stacked CNNs do not maintain a temporal relationship between them, and the composition of knowledge is still decoupled between the stacks.

2.3.1 The Recurrent Attention Model (RAM)

The Recurrent Attention Model (RAM)[80] is currently one of the most innovative frameworks for visual attention, although little explored in subsequent literature. RAM works by iteratively building a latent space through multiple glimpses, similar to biological visual attention. Therefore, the input image is scanned through the cropping and resizing of small patches; then, feature extraction is employed to detect the most salient information. This information is iteratively stored in a latent space, which provides the knowledge to perform the task at hand. The location of each glimpse is determined by a policy learned through REINFORCE [130] in a reinforcement learning setup.

Although presenting innovative concepts, RAM was mainly concerned with concept proof for classification tasks in the MNIST dataset, unable to deal with more complex tasks. Its architecture is composed of four networks, as shown in Figure 2.7, which comprehends the glimpse network f_g, the core network f_h, the location network f_l, and the action network f_a. Each network encapsulates important characteristics, such as extracting visual information, integrating the extracted information, and generating the next focus location and predictions.

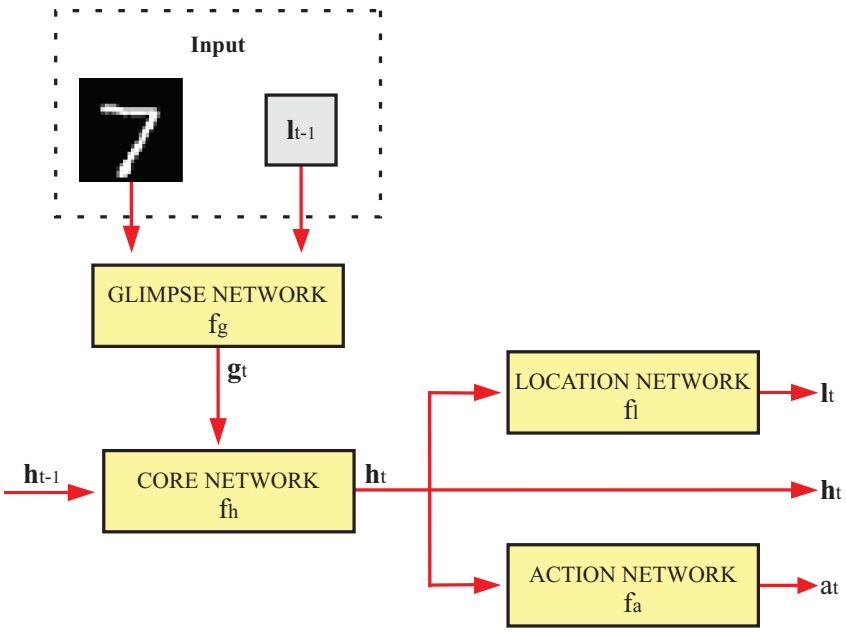


Figure 2.7: The RAM architecture is composed of four networks. The glimpse network f_g extracts visual features from the input image, given a focus location $\mathbf{l}_{t-1}$; the core network f_h integrates these features in an informative latent space $\mathbf{h}_t$; the location network f_l generates the image focus location $\mathbf{l}_t$ for the subsequent iteration; the action network f_a generates the class prediction a_t.

The glimpse network f_g represents the perceptual system and is responsible for extracting meaningful information from the input images. The glimpse network comprises

a glimpse sensor that implements attention and several linear layers to generate the latent space. First, the glimpse sensor receives an image $\mathbf{x}_t$ and a focus location $\mathbf{l}_{t-1}$ as input. The attentional system discretely selects several image patches centered at the focus location $\mathbf{l}_{t-1}$; each patch is selected with an increased resolution. This process builds a pyramidal-like structure, further reduced by either the max or average pool function. Then, all patches are concatenated to form the retina representation $\rho(\mathbf{x}_t, \mathbf{l}_{t-1})$. Finally, the glimpse network concatenates ρ_t and $\mathbf{l}_{t-1}$ to produce the glimpse feature vector $\mathbf{g}_t$. The entire process is mathematically described as follows

$$\mathbf{h}_l = \text{Rect}\left(\text{Linear}\left(\mathbf{l}_{t-1}\right)\right), \tag{2.26}$$

$$\mathbf{h}_g = \text{Rect}\left(\text{Linear}\left(\rho\left(\mathbf{x}_t, \mathbf{l}_{t-1}\right)\right)\right), \tag{2.27}$$

$$\mathbf{g}_t = \text{Rect}\left(\text{Linear}\left(\mathbf{h}_g\right), \text{Linear}\left(\mathbf{h}_l\right)\right), \tag{2.28}$$

with $\mathbf{h}_g$ and $\mathbf{h}_l \in \mathbb{R}^{128}$, and $\mathbf{g}_t \in \mathbb{R}^{256}$. Also, the linear layers and the rectifier nonlinearity are defined by

$$\text{Linear}\left(\mathbf{x}\right) = \mathbf{W}\mathbf{x} + \mathbf{b}, \tag{2.29}$$

$$\text{Rect}\left(\mathbf{x}\right) = \max\left(\mathbf{x}, 0\right), \tag{2.30}$$

where $\mathbf{W}$ is the weight matrix, and $\mathbf{b}$ is the bias vector.

The core network f_h is responsible for storing the information across multiple glimpses. Therefore, it receives the glimpse feature vector $\mathbf{g}_t$ and the previous internal state $\mathbf{h}_{t-1}$ as input at every time step t. Through linear layers, it outputs the current internal state $\mathbf{h}_t$, which condenses all the sequential information provided by the glimpse network. The core network is mathematically described by

$$\mathbf{h}_t = \text{Rect}\left(\text{Linear}\left(\mathbf{h}_{t-1}\right) + \text{Linear}\left(\mathbf{g}_t\right)\right), \tag{2.31}$$

where $\mathbf{h}_t \in \mathbb{R}^{256}$. For simple experiments in the MNIST dataset, f_h is originally composed of linear layers; however, in dynamic experiments, the linear layers are substituted by LSTM layers.

The location network f_l is responsible for generating the focus location $\mathbf{l}_t$ for the subsequent input image. Therefore, it uses the internal state $\mathbf{h}_t$ to parameterize the mean μ_t for a Gaussian distribution p with two dimensions and a fixed standard deviation of 0.05. The mathematical definition is

$$\mu_t = \text{Linear}(\mathbf{h}_t), \tag{2.32}$$

$$\mathbf{l}_t \sim p(\cdot|\mu_t), \tag{2.33}$$

where $\mathbf{l}_t$ is the reparameterized sample, which allows the network to be differentiable.

The action network f_a is responsible for predicting the class a_t, which is the architecture's ultimate goal. Therefore, the authors employed the Softmax function

$$a_t = \text{argmax}\left(\frac{e^{\text{Linear}(\mathbf{h}_t)}}{Z}\right), \tag{2.34}$$

where Z is the normalizing constant.

Considering that RAM uses a hard attention mechanism, i.e., only parts of the image are selected for processing, optimization via supervised learning is not feasible. Hence, reinforcement learning strategies are used in training. This RL setup is defined as an instance of a Partially Observable Markov Decision Process (POMDP), in which the true state of the environment is unobserved; therefore the model needs to learn a stochastic policy $\pi\left((l_t, a_t)\,|s_{1:t}; \boldsymbol{\theta}\right)$ with the parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_g, \boldsymbol{\theta}_a, \boldsymbol{\theta}_h\}$ that maps the environment history $s_{1:t} = \{x_1, l_1, a_1, ..., x_t, l_t, a_t\}$ to a distribution over the action in time step t, restricted by the sensor. In this sense, the glimpse is considered the agent, the environment is the whole image presented to the glimpse sensor, and the rewards are defined according to the success in the classification. After executing a classification action, the agent receives a reward r_{t+1} and a new observation – a patch of the input image. The goal is to maximize the sum of the reward signal which is generally sparse and delayed, therefore, the return is $G = \sum_{t=1}^{T} r_t$. For example, the agent receives $r_t = 1$ from the ground-truth values if the class is classified correctly after T time steps, and 0 otherwise.

The architecture parameters $\boldsymbol{\theta}$ are optimized by maximizing the total reward when the agent interacts with the environment. The agent's policy, in combination with the environment's dynamics, produces a distribution over the possible iteration sequences $s_{1:N}$, and the goal is to maximize the return under that distribution via

$$J(\boldsymbol{\theta}) = \mathbb{E}_{p(s_1:T;\boldsymbol{\theta})}[G], \tag{2.35}$$

where $p(s_1 : T; \boldsymbol{\theta})$ depends on the policy.

Maximizing $J(\boldsymbol{\theta})$ is not trivial because it involves an expectation about high-dimension iteration sequences, which may involve an unknown environment dynamics. However, it is possible to obtain an approximation of the gradient given by the REINFORCE algorithm [130] as follows

$$\nabla J(\boldsymbol{\theta}) = \frac{1}{M}\sum_{i=1}^{M}\sum_{t=1}^{T} \nabla \log \pi\left(u_t^i | s_{1:t}^i; \boldsymbol{\theta}\right)(G_t^i - b_t), \tag{2.36}$$

where s^i are the sequences obtained by running the current agent policy π_θ for $i = 1, ..., M$ episodes, G_t^i is the accumulated reward obtained after executing the action, and b_t is the baseline value, which reduces the variance for the gradient updates. The baseline value b_t depends on sequence $s_{1:t}^i$ via the hidden state h_t^i but not directly of the action u_t^i. As a result, the algorithm increases the log-probability of actions that generate a high cumulative reward and diminishes the probability of actions that generates a low cumulative reward. Also, the baseline loss aims to reduce the mean squared error between the baseline value b_t and the return G_t^i.

The REINFORCE algorithm [130] allows training the agent when the best actions

are unknown, and the learning signal is provided only through reward. It is possible not to know a priori which sequence of glimpses provides more information about an unknown image, but the total reward at the end of the episode indicates whether the sequence was good or bad. For image training, the label is known, making it possible to directly optimize the policy to generate the correct label associated with a training image at the end of an observation sequence. This is achieved by maximizing the conditional probability of the ground-truth label given the image observations, that is, maximizing $\log[\pi(a_T^*|s_{1:T};\boldsymbol{\theta})]$, where a_T^* is the ground-truth image. Following this approach, the action network is optimized by the cross-entropy loss; and the gradients are propagated through the core and glimpse network. The location network is therefore detached and trained only by the REINFORCE algorithm, producing a hybrid supervised loss [80].

2.4 Final Considerations

Visual odometry methods can be built in several different ways, depending on the project's goals and restrictions. In this work, we are mainly interested in building a cost-effective method to learn from the input data. Therefore, the monocular setup will be utilized due to a) cost reduction by decreasing the required sensors; b) stereo setups degenerate to monocular at great distances; c) stereo setups require constant calibration due to the agent's motion and shaking.

For visual odometry tasks, RAM [80] can be classified primarily as a direct method due to operating over the pixel values. However, the attentional glimpse acts as a feature selector, where the extracted patches can be seen as regions of interest (i.e., keypoints). Therefore, RAM can access all the image information and work directly with the raw data while keeping its low-cost nature by selecting only the regions of interest. Further, reinforcement learning allows the agent to learn a robust policy, enabling the architecture to track motion, avoid specific object/regions in the scene, and be invariant to image resolution. These characteristics are vital to visual odometry and constitute the primary source of error.

However, RAM is mainly a conceptual architecture, being tested only for simple classification tasks on the MNIST dataset. The visual odometry problem is significantly more complex; the input images have a higher resolution, consisting of many mobile objects, illuminance variation, shadows, and other non-linearities. The 6-DoF regression is also quite tricky; the minimum error in the angle at the start results in a substantial accumulating error at the end. Hence, extending RAM requires the addition of spatial, temporal, and contextual elements; the implementation of robust RL algorithms such as Proximal Policy Optimization (PPO) [111]; and the elaboration of alternative reward functions.

Chapter 3

Related Work

This chapter presents the most relevant literature related to our work. The presented methods concern mainly the use of supervised deep learning in visual odometry. Further, a section is dedicated to attentional methods in the VO field. Finally, we present some consideration on reinforcement learning.

3.1 Visual Odometry and Deep Learning

The first visual odometry solution implemented with a deep learning method was proposed by Konda et al. [63] in 2015. Their model implemented an end-to-end CNN architecture to estimate motion direction and velocity from raw stereo images, as shown in Figure 3.1. The architecture is composed of several CNN layers followed by a fully connected layer. The network's output consisted of a softmax activation function; therefore, the VO problem was formulated as a classification problem – due to limited data, a regression approach has not provided satisfying results. The first CNN layer was initialized by a synchrony/depth autoencoder [62], which increases the model's accuracy and provided better filters. The use of depth information during the initialization reduced overfitting, and it was fundamental to solve ambiguity issues when determining changes in the agent's direction. The training was performed on the KITTI [42] dataset and consisted of a 5-frame stereo sequence as input and a pre-computed discretized vector of velocities and direction changes as labeled data. The results were promising concerning the velocity and direction predictions; however, there were significant drift errors – possibly due to the limited model's capacity, since visual data usually requires several CNN layers to capture the scene dynamics.

The first method capable of performing VO with monocular images was proposed by Mohanty [81] in 2016. Their method estimated the trajectory by computing the transformation matrix between two frames based on the extraction of high-level features. The architecture is composed of a fully-connected CNN partially inspired by AlexNet [65], which receives a pair of monocular images ($\mathbf{I}_t$, $\mathbf{I}_{t+1}$) corresponding to subsequent time periods, and pre-processed labeled ground-truth values representing the differential displacement in pose $(\Delta x, \Delta y, \Delta\theta)$ at each time instant, as shown in Figure 3.2. Unlike previous techniques, this solution is formulated as a regression problem capable of learning

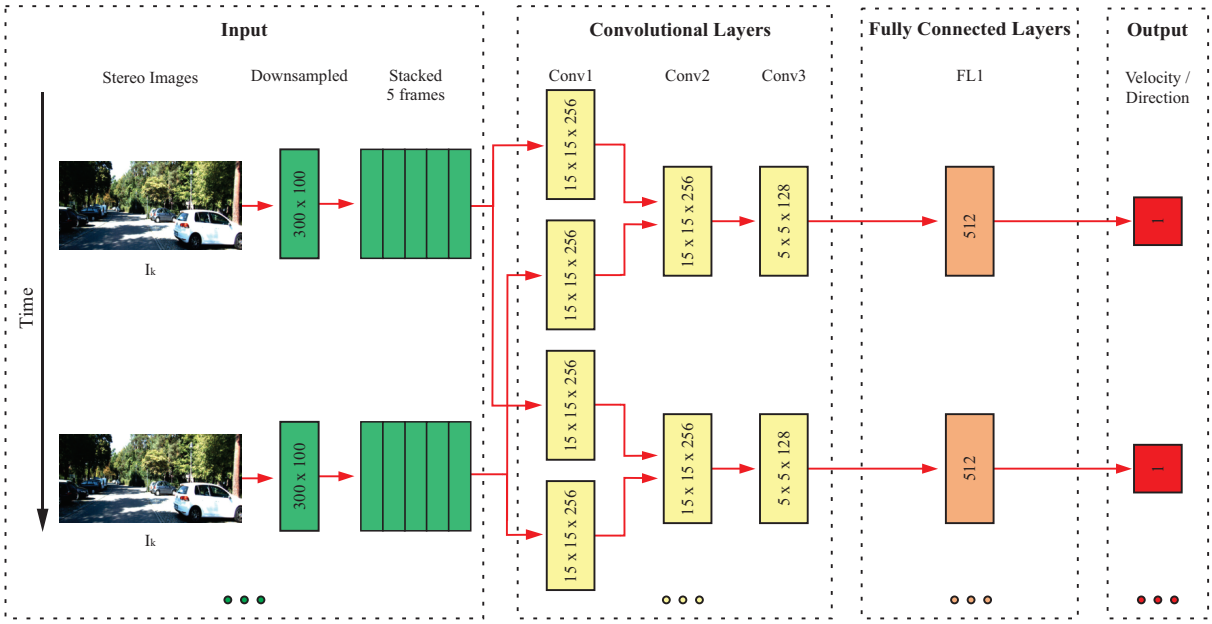


Figure 3.1: The architecture proposed by Konda [63] is composed of convolutional layers followed by fully connected layers. The architecture can predict the agent's velocity and direction.

the camera's intrinsic parameters and pose. It also provided encouraging results for predicting absolute scale using only one camera. The training was carried out on the KITTI [42] dataset using the L2 loss function. The results in known environments (i.e., seen during training) were accurate. However, results in unknown environments performed poorly, even when the FAST feature detector was incorporated during training. The poor performance can be explained by the architecture's nature, which lacks the structures to capture temporal/sequential relationships.

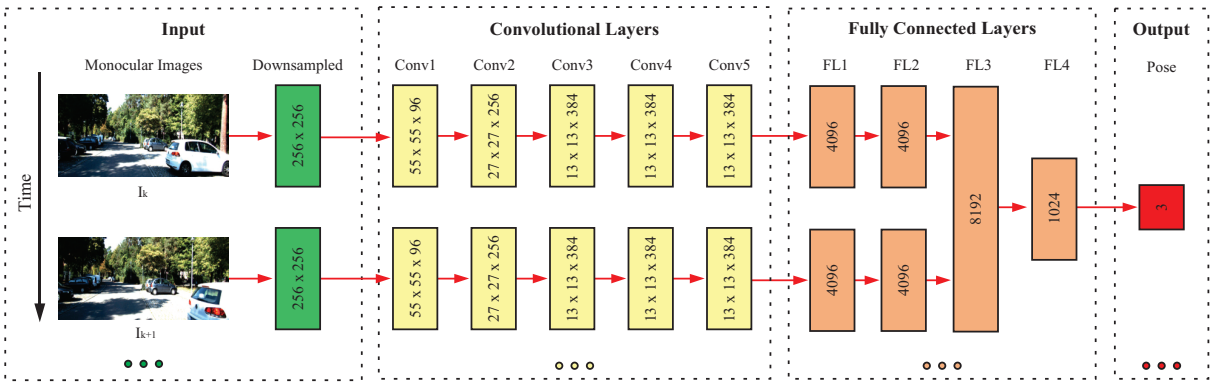


Figure 3.2: The architecture proposed by Mohanty [81] is composed of convolutional and fully connected layers. The architecture outputs the displacement in pose $(\Delta x, \Delta y, \Delta\theta)$.

Recovering the world scale is a big issue for monocular VO due to the ill-posed problem of estimating the depth from a single image – stereo VO can reliably compute the depth by triangulation but require careful calibration. To address this issue, Yin et al. [135] proposed an architecture able to recover the scale by estimating the depth from the input monocular images. The authors proposed to use a modified ResNet CNN-based archi-

tecture and conditional random fields to estimate the depth. Tang et al. [119] proposes a Geometric Correspondence Network, which consists of a CNN trained together with an RNN to detect the keypoints' location and generate descriptors in the same architecture. The proposed GCN works by warping points from a source frame to a target frame. The convolutional networks were implemented based on ResNet. Brahmbhatt et al. [8] proposes an aware geometry method, which learns the camera's global position based on data-driven map representation with geometric constraints between two images. However, this method only works for already known environments.

Other methods aim to perform VO by extracting the optical flow from the input images. The Flowdometry architecture proposed by Muller and Savakis [86] extracts the optical flow from monocular RGB images, as shown in Figure 3.3. Their architecture is trained in an end-to-end manner, without any algorithm to pre-process the frames used in P-CNN VO [22]. Flowdometry is entirely based on the FlowNetS [36] architecture due to the better results it presents in natural scenes than FlowNetC. Therefore, the pipeline's first stage computes the optical flow directly from pairs of monocular images using the original FlowNetS; the second stage computes the incremental changes in angle and displacements using a re-purposed version of FlowNetS, which replaced the input from a six-channel image to a two-channel optical flow, and the modification in the output layer to allow the pose regression. The entire solution is based on CNN layers and trained from scratch without pre-trained weights, as seen in P-CNN VO [22]. The results seem promising, considering the possibility of operating in real-time due to the avoidance of the high-cost computation of extracting optical flow by the Brox algorithm.

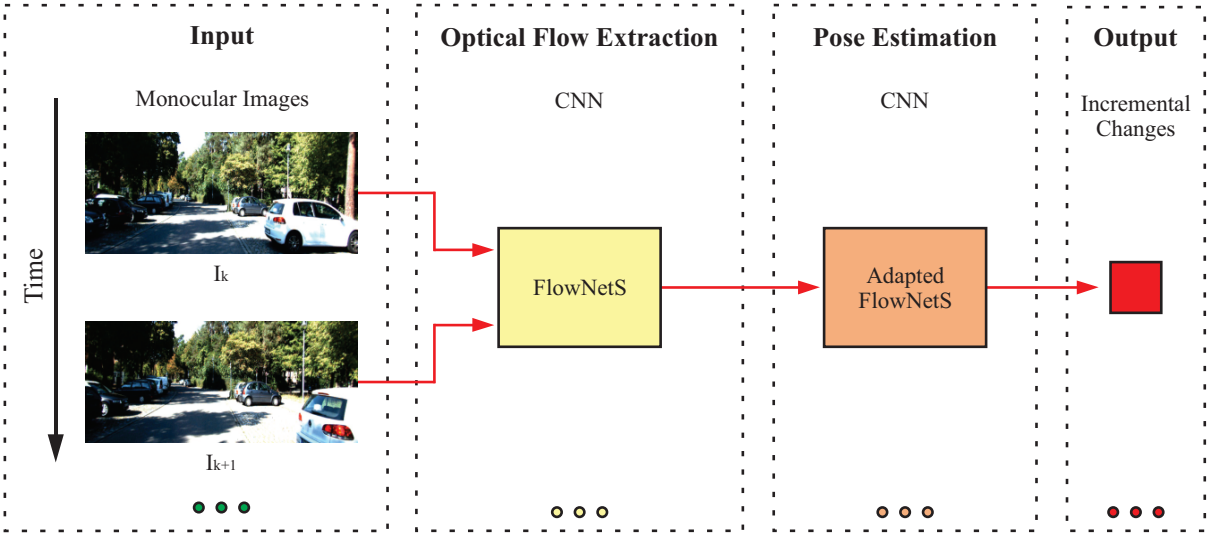


Figure 3.3: The Flowdometry architecture proposed by Muller [86] extracts the optical flow based on the FlowNet and outputs the pose's displacements.

Peretroukhin and Kelly [95] proposed the DPC-Net architecture, which aims to integrate the representation capabilities of deep neural networks with the efficiency of geometric and probabilistic algorithms. Therefore, DPC-Net implements a CNN-based architecture to learns the corrections for the pose estimator. In this sense, the authors proposed novel loss functions based on the Lie groups, balancing the translational and rotational

errors. DPC-Net was tested on the KITTI dataset and improved the results compared to the baseline model, mitigating poor sensor calibration and environmental factors such as lens distortion. Zhao et al. [137] proposes the L-VO architecture, which predicts the 6-DoF pose from 3D optical flow for a monocular VO. The loss function employs a Bivariate Gaussian model. The optical flow is computed by the FlowNet architecture and the depth map by the DepthNet architecture. The dense 2D optical flow and the depth map are used to generate a 3D optical flow, further consumed by the L-VO. The architecture is tested on the KITTI dataset and presented better results than monocular geometric methods and ESP-VO; however, the KITTI images were downsampled to 320x96 pixels, degrading the performance.

Instead of explicitly extracting the optical flow, Wang et al. [126] proposed the DeepVO, which is an end-to-end monocular architecture capable of extracting features and modeling the sequential dependence right from the input raw images. DeepVO consists of a CNN that extracts the interesting features from the monocular images; these features are further processed in an LSTM network, which can infer temporal dependencies and, consequently, determine a more robust 6-DoF pose, as shown in Figure 3.4. DeepVO does not need the camera parameters nor prior knowledge to determine the absolute scale, considering the ground-truth values are provided during training. Even parameter fine-tuning, found in most VO pipelines, is avoided since the model is trained in an end-to-end manner. The experiments were performed on the KITTI dataset, employing the MSE of all positions and orientations as the loss function. The results have shown that DeepVO can retrieve the absolute scale similar to stereo methods. Compared to [63, 81], the better results are explained mainly by the use of recurrent layers and the deeper CNN stage. However, the results were still inferior to geometric methods, making the authors consider DL as a complementary tool.

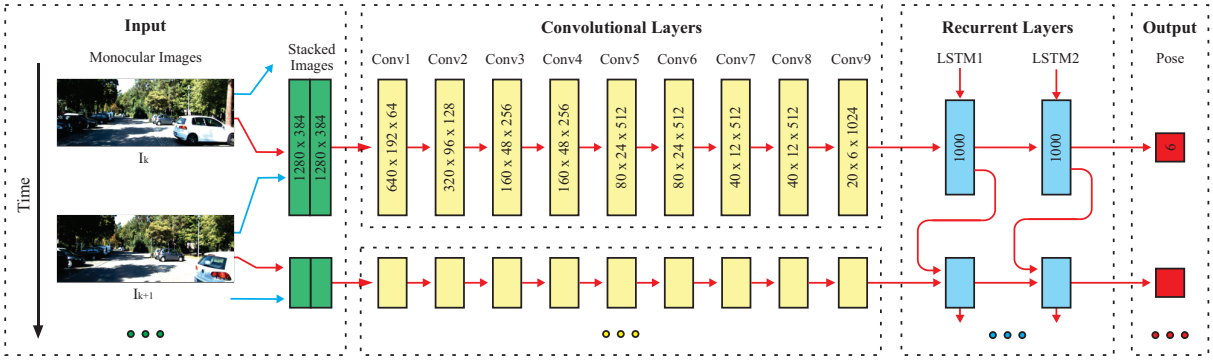


Figure 3.4: The DeepVO architecture proposed by Wang [126] is composed of several convolutional layers to extract the visual features from the input images. Also, the architecture implements LSTM layers to regress the 6-DoF pose.

Long-term navigation still poses a significant challenge for visual odometry due to the accumulating drift error – robotic systems are not able to operate for a long time in real-world scenarios without accumulating significant error. One solution is to employ semantic information in the VO pipeline. In this context, Valada et al. [125] proposed the VLocNet architecture, which is capable of estimating the 6-DoF pose in a monocular setup. Their architecture fuses relative and global RCNN-based architectures to improve accuracy. The

relative architecture is used to smooth the VO path, while global architecture reduces the drift error. Although the use of recurrent networks has improved the VO accuracy in recent years, these networks still have limited capacity for storing long-term data – an alternative is to increase storage by using external memories.

Training deep neural networks for VO applications requires a vast amount of computational resources, limiting its application in constrained contexts (e.g., mobile phones, MAV). Several alternatives tried to compress the neural networks (e.g., quantization and pruning) at the cost of reducing accuracy. An emerging approach is to employ Knowledge Distillation (KD) to train two networks jointly. KD enables the transfer of knowledge from a teacher network to student networks, similar to an imitation learning model. Using these concepts, Saputra et al. [103] proposed to distill knowledge from a pose regressor – up to that point, KD had only been applied to classification problems.

Learning methods usually perform VO by regressing the 6-DoF pose given a pair of images. However, these methods do not consider essential transformation properties, which can improve the model's accuracy. To tackle this issue, Wang et al. [129] proposed a monocular VO that implements a novel cost function based on mathematical properties of group homomorphisms. Mainly, the authors aim to guarantee the properties of closure, identity element, and inverse element. Although these new cost functions were implemented in a supervised learning context, they can also be used for unsupervised methods. The authors validated their architecture on the KITTI dataset, where they implemented two versions: using only L2 loss and the novel homomorphism loss. The results present by the novel loss function outperform the usually used L2 loss. Also, the authors proposed to reduce the drift error by estimating drivable paths using semantic segmentation. The idea is that pose estimation should be consistent with the observed path.

Directly learning the 6-DoF pose from a scene represents a great difficulty for convergence due to the accumulative errors. One alternative is to incrementally increase the task difficulty while learning the pose so that the network gradually converges to a good solution. In this context, Saputra et al. [105] proposes the first Curriculum Learning (CL) architecture, called CL-VO. It aims to learn scene geometry for monocular VO using a bio-inspired learning paradigm, which gradually increases the task's difficulty. Their architecture is composed of extraction and regression stages; the extraction stage comprehends the use of FlowNet to extract the optical flow and generate a latent space; the second stage is composed of two layers of LSTM, which regresses the 6-DoF pose. Curriculum learning is defined as the composite loss function, which penalizes the pose errors for a small bounded window. Therefore, CL-VO incrementally learns the scene geometry by gradually giving more weight to the composite loss during training. The authors compared their method with others that utilize reverse curriculum and no curriculum; as expected, the network presented more difficulty to converge when other learning approaches were employed, validating the CL-based approach.

Real-time VO is still a complicated task. Several methods extract optical flow; however, extracting the optical flow is a costly procedure. In this context, Guo et al. [48] proposed the LightVO, which uses a CNN-based architecture called TVNet to extract the optical flow. The pose estimation is achieved by DenseNet, which reduces the amount of parameters and training time. Training was perform using the KITTI dataset, and

results proved it to be at least 50% faster than similar methods, such as Flowdometry and P-CNN. However, the rotation estimation errors were corrected by employing IMU readings combined with a Kalman Filter.

Segmenting images in drivable regions brings scene understanding, which can be further explored in odometry methods. In this context, Holder et al. [54] proposed a CNN architecture able to predict the path taken by the vehicle in off-road environments. Their architecture segments the path that the driver would likely take and maps it into the image. The authors proposed three CNN-based architectures (i.e., SegNet, U-Net, and FCN). The U-Net architecture achieved the best results in terms of memory usage and speed. Also, the authors built their own dataset.

Feature-based methods deliver poor results in low-texture environments; even direct learning methods can present difficulties to converge in such scenarios. A common alternative is to enhance the input images before passing them to the learning system. Therefore, Yan et al. [134] proposed an RCNN architecture able to combine monocular RGB images with their edge-enhanced correspondence. Their architecture enhanced the input images with the Canny Detector and passed them to a CNN+LSTM architecture to regress the 6-DoF pose. The achieved results showed an improvement in performance compared to monocular VISO, but still worse than stereo geometric methods. In general, VO systems have several issues with illumination variations and high-dynamic-range environments (HDR). The sensors are responsible for such problems due to assumptions such as brightness constancy. To tackle HDR scenarios, Gomez-Ojeda et al. [44] proposed two architectures based on CNN+LSTM to enhance the entire VO image sequence. Their architecture comprehends an encoder composed of convolutional layers initialized with VGGNet weights, LSTM recurrent layers to guarantee sequentially consistent images, and a CNN-based decoder that outputs the improved image. The VO results with the corrected images showed a slight improvement compared to the original images; visual inspections indicate that the corrected images presented better gradient information.

Similarly, direct methods rely on assumptions such as photometric consistency; however, in practice, they are usually violated. To attenuate this issue, Clement and Kelly [19] trains a CNN-based encoder-decoder network to predict the Canonical Appearance Transformations (CAT) given input frames under different illumination settings. The authors chose the U-Net architecture to perform the image compression and decompression, given skipped connections, which allows the network to preserve information. The network can generate a canonical image as output, which improves the robustness against illumination variations due to camera response and environment dynamics, therefore improving the visual odometry accuracy.

The presented supervised deep learning methods are summarized in Table 3.1. Also, we reported other VO methods, which add mainly technological contributions. Many other VO methods use unsupervised and self-supervised learning. However, they are not in the scope of this work. In the table, we summarize relevant aspects of a VO DL-based model such as network architecture, loss function, input and output data, and dataset used for training.

Method	Year	Arch	Goal	Input	Output	Loss	Dataset
Konda and Memisevic et al. [63]	2015	CNN	End-to-end stereo VO	Stereo images	Velocity, direction	-	KITTI
P-CNN VO [22]	2016	CNN	Mono VO	Optical flow	Disp. camera, euler orient.	RMSE	KITTI
Mohanty et al. [81]	2016	CNN	End-to-end mono VO	Mono images	Diff. changes $(\Delta x, \Delta y, \Delta\theta)$	L^2	KITTI
VINet [18]	2017	CNN, LSTM	Visual-inertial odometry	Mono images, IMU	3D trans, 4D orient. quart.	L^2	EuRoC, KITTI
Flowdometry [86]	2017	FlowNetS	Mono VO	Optical flow	Inc. changes (ang, displac.)	-	KITTI
DeepVO [126]	2017	CNN, LSTM	End-to-end mono VO	Mono images	6-DoF pose	MSE	KITTI
Yin et al. [135]	2017	ResNet	Depth estimat. for mono VO	Mono images	Depth map	MSE, Huber	KITTI
Haarnoja et al. [49]	2017	CNN	State estimat. w/ KF for VO	Mono images	KF state estimation	-	Synthetic, KITTI
DeepTAM [138]	2018	CNN	Tracking and mapping VO	Stereo images	L^2	6-DOP pose	SUN3D, SUNCG
EndoVO [123]	2018	Inception, LSTM	Endoscopic VO	Mono images	6-DoF pose	L^2	Own
Sun-BCNN [94]	2018	CNN	Inferring sun direction	Mono image	Sun direction	Cosine distance	KITTI
VLocNet [125]	2018	ResNet, CNN	End-to-end global pose VO	Mono images	6-DoF pose	L^2	7-Scenes, Camb. Land.
GCN [119]	2018	ResNet, RCNN	Keypoint detection for VO	RGB images	Keypoint location	L^2	KITTI, TUM
ESP-VO [127]	2018	CNN, LSTM	End-to-end probabilistic VO	Mono images	6-DoF pose, covariance	MSE, cond. probability	KITTI. EuRoC
Gomez-Ojeda et al. [44]	2018	CNN, LSTM	Image enhancing for mono VO	Mono images	Enhanced image	Log RMSE, SSIM	Adapted, Synthetic
MapNet [8]	2018	PoseNet	Geometry-aware VO	RGB images	6-DoF pose	Global-local pose dist.	7-Scenes, RobotCar
Holder and Breckon et al. [54]	2018	SegNet, U-Net	Off-road path segmentation	Mono images	Segmented path	MSE	Own
Clement and Kelly et al. [19]	2018	U-Net	Image enhancing for mono VO	Mono images	Enhanced image	Squared L^2	ETHL KITTI
DPC-Net [95]	2018	CNN	Stereo VO	Stereo images	6-DoF pose	Lie-group corrections	KITTI
L-VO [137]	2018	FlowNet, DepthNet	End-to-end mono VO	3D Optical flow	6-DoF pose	L^2, Bivariate Gaussian	KITTI
GFS-VO [131]	2018	FlowNet, ConvLSTM	Context aware mono VO	Mono images	6-DoF pose	L^2	KITTI, EICL-NUIM
Chen et al. [13]	2019	FlowNet, IONet	End-to-end mono VIO	Mono images, IMU	6-DoF pose	-	KITTI, EuRoC, PennCOS.
Lin et al. [71]	2019	ResNet, LSTM	Global-relative mono VO	Mono images	6-DoF pose	Cross trans. const., MSE	7-Scenes, KITTI
Xue et al. [132]	2019	CNN, LSTM	Attentional mono VO	Mono images	6-DoF pose	L^2-local, L^2-global	KITTI, TUM
Ruan et al. [101]	2019	FlowNet, CNN	End-to-end VO	Mono/stereo images	6-DoF pose	MSE	KITTI
Saputra et al. [103]	2019	ESP-VO, CNN, LSTM	Knowledge distillation VO	Mono images	6-DoF pose	Attentive Imitation	KITTI, Malaga
InertialNet [73]	2019	FlowNet2, CNN	End-to-end mono VIO	Mono images	IMU data	MSE	EuRoC, own
3DC-VO [64]	2019	CNN	End-to-end mono VO	Mono images	6-DoF pose	MSE	KITTI, own
Wang et al. [129]	2019	CNN	New homomorphism losses	Mono images	6-DoF pose	L^2, Homomorphism losses	KITTI
CL-VO [105]	2019	FlowNet, LSTM	End-to-end curriculum learning VO	Mono images	6-DoF pose	Bounded pose regression loss	KITTI, Malaga, own
Chen et al. [14]	2019	FlowNet, LSTM	End-to-end monocular VO	Mono images	6-DoF pose	MSE	KITTI
LightVO [48]	2019	TVNet, DenseNet	End-to-end mono VIO	Optical flow, IMU	6-DoF pose	-	KITTI
Yan et al. [134]	2019	CNN, LSTM	Image enhancing for mono VO	Mono enhanced images	6-DoF pose	MSE	KITTI
Teixeira et al. [120]	2020	SfmLearner, GeoNet, LSTM	Underwater VO	RGB images, IMU	6-DoF pose	MSE, quaternion distance	Own
DeepTIO [104]	2020	CNN, LSTM	Thermal-inertial odometry	Thermal images, IMU	6-DoF pose	Huber	Own
DeepPCO [128]	2020	CNN, FlowNet	Point-cloud odometry	Depth images	6-DoF pose	MSE	KITTI

Table 3.1: A summary of supervised deep learning methods.

3.1.1 Attentional Methods

In recent years, several methods started to use concepts of attention in their pipeline. Attention is mainly employed to filter out unnecessary information and reduce computational resources to perform VO in complex scenes. Most methods utilize attention to weight either the input image or the latent space inside the architecture. When employed in the inputs, the attention mechanism allows the network to select meaningful informa-

tion from the whole image, which is a complicated task for CNN-based architectures.

In 2020, Damirchi et al. [24] explored the concept of self-attention to extract meaningful features in complex scenarios, which usually have many moving objects and low texture. The authors were interested in retrieving global-scale information from the input images, which is complicated with only CNN networks because they provide local-scale information. Therefore, an alternative is to use self-attention, which correlates all the input image pixels, allowing the network to pool information from different areas. The self-attention block is placed right after the CNN layers and before the LSTM layers. The experiments were performed on the KITTI dataset and showed a reduction in the drift error compared to DeepVO [126]. Therefore, using self-attention for spatial features is a viable and simple option to improve visual odometry by generating a more representative latent space.

Also, Kuo et al. [66] proposed the DAVO architecture, which is a dynamic attention-based visual odometry composed of two attentional networks. Their first network can generate semantic masks for determining the weights that each piece of the input image should have, while the second network uses a squeeze-and-excite attentional block. The authors also provided several ablation studies to prove the validity of their method. The experiments were conducted on the KITTI dataset and showed the superior performance of DAVO compared to other state-of-the-art methods.

Equally important in VO is to be able to save and retrieve meaningful information across multiples frames. Therefore, Xue et al. [132] proposed to add memory and refining components inside the VO pipeline. The use of memory enables the architecture to maintain global information through selection criteria. The refining components use a spatial-temporal attention mechanism to improve accuracy. The methods achieve impressive results in environments where traditional methods fail (e.g., low texture, sudden motion variation). The authors tested their method in the KITTI and TUM-RGBD datasets. Further, Xue et al. [133] proposed an adaptive memory block, which saves and refine the information from local-scale to global-scale. Therefore, the architecture can store and process long-term dependencies to refine the local pose estimation. The experiments were conducted on the KITTI dataset and had shown that their methods could generate predictions similar to classic methods.

Attentional concepts are also employed to support the salient feature extraction from the input images. Liang et al. [70] proposes the SalientDSO architecture, which applies attention in the Direct Sparse Odometry (DSO) [30] algorithm. The proposed method runs in two distinct modules, the first one detects the visual salience using SalGAN, and the second one performs visual odometry with DSO. The drift error has substantially decreased compared to the original DSO due to the improved sampling from salient points instead of randomly selected. However, SalientDSO has not presented significant improvement in sequences where there are insufficient salient points. Chen et al. [14] also proposed an end-to-end CNN+LSTM salient-feature attention and context-guided networks for robust visual odometry. Their architecture can be trained using only monocular images and aimed to decouple rotational and translational motion. The first stage selects salient features using a pre-trained VGG in a FlowNet backbone; the second stage regresses the pose in two parallel LSTM architectures, decoupling the rotational and translational mo-

tion. The experimental results in the KITTI dataset showed better performance than state-of-the-art methods such as monocular VISO, ESP-VO, and DeepVO; however, the impact was caused mainly by decoupling the motion in two independent LSTM.

3.2 Final Considerations

The visual odometry field has seen a massive increase in the number of architectures and publications in recent years, mainly with the advent of deep learning. Geometric methods still present the best results in restricted and well-controlled scenarios, while deep learning methods already achieve good results in open-world and complex environments. However, deep learning methods demand a high computational cost and the creation of large and representative datasets. One recent alternative is performing VO with unsupervised methods because they do not require data labeling. Despite the benefits, these architectures are substantially large and costly to train due to the necessity of depth maps currently being generated by Generative Adversarial Networks (GAN) [1, 34].

Therefore, the creation of lightweight and efficient methods is fundamental to the field. The massive increase in robotic applications and mobile devices will require power-efficient algorithms. Also, the capacity to collect only the necessary information is essential due to the highly dynamic environment. In this sense, one way forward is to bring innovative concepts to the field. Several methods have appeared, using both supervised and unsupervised learning, which have presented important concepts. Still, we have not found any method that implements reinforcement learning in any part of the pipeline. RL applied to visual odometry tasks allows the architecture to learn a robust policy to deal with complex scenarios, therefore mitigating the drift error. Also, the architecture becomes efficient by selecting only the necessary input data.

Chapter 4

Materials and Methods

This chapter presents the materials and the methods used to develop this work. The datasets used to train the model will be detailed, including the metrics used to evaluate the results' quality. We specify the software and hardware technologies used in the experiments. Also, we present the methodology employed to build the architecture and reach the final results.

4.1 Materials

4.1.1 Popular and Created Datasets

In the early years, visual odometry datasets were created to leverage simple algorithms for localization, mapping, and odometry systems. Visual data acquisition started inside the universities, which collected data by agents (e.g., car, robot, human) circulating on their campuses or in controlled scenes. However, academic environments presented limited complexity, making it necessary to acquire data in larger real-world scenarios. The popularization of autonomous navigation systems has further requested the construction of diverse datasets to address the new challenges caused by complex environments, such as many moving objects, occlusions, low visibility, noisy sensors, and long-term functioning. As a summary, Table 4.1 presents the most popular datasets for visual odometry, including their main features.

In this work, we mainly used the KITTI (Karlsruhe Institute of Technology and Toyota Technological Institute) dataset [42], a vision benchmark suite developed to promote the creation of new algorithms for the robotic and computer vision fields. The dataset was built in 2013 and remained one of the most popular datasets for autonomous navigation in an outdoor environment. The dataset consists of 22 sequences (total length of 39.2 km) of real-world traffic data captured by a car moving across urban and rural areas in Germany. The dataset is divided into several categories (e.g., Road, City, Residential, Campus, Person), contemplating a wide variety of static and mobile objects. These objects are annotated in 3D bounding boxes and classified in the most seen classes (e.g., Car, Van, Truck, Pedestrian, Person, Cyclist, Tram, Misc). Stereo cameras captured high-resolution grayscale and RGB images. A Velodyne 3D laser scanner captured the point cloud, and an IMU/GPS navigation system captured the positional/inertial data.

Dataset	Year	Carrier	Enviro.	Camera	Other sensors	Ground-truth	Length
New College[113]	2009	Seg way	Outdoor	Stereo gray 512x384 @20 Hz, Panoramic RGB 384x512 @3 Hz	1x IMU, 1x GPS @5Hz, 2x Laser @75 Hz	GPS, wheel odometry @28 Hz	3 seqs, 2.2 km
TUM RGB-D[115]	2012	Hand held, robot	Indoor	Mono RGB-D 640x480 @30 Hz	Depth sensor, accelerometer @500 Hz	Motion capture 8x cameras @100Hz, acc. $<$ 1 mm	39 seqs
KITTI [42]	2013	Car	Outdoor	Stereo RGB/gray 1392x512 @10 Hz, global shutter	1x Laser @10 Hz, 1x IMU/GPS @100Hz, soft. sync.	IMU/GPS, acc. $<$ 10 cm	22 seqs, 39.2 km
ICL-NUIM [50]	2014	Hand held	Synth. Indoor	Mono RGB-D 640x480 @30 Hz, rolling shutter	Depth sensor	Provided by simulator	8 seqs, 58.46 m
Malaga Urban [7]	2014	Car	Outdoor	Stereo RGB 1024x768 @20Hz	5x Laser, 1x IMU @100 Hz, 1x GPS @1 Hz, soft. sync.	GPS, low accuracy	15 seqs, 36.8 km
TUM Mono [32]	2016	Hand held	Indoor/ Outdoor	Mono gray 1080x1024 @20-50 Hz, global shutter	-	Loop closure, low accuracy	50 seqs, 100 min
EuRoC MAV [10]	2016	MAV	Indoor	Stereo gray 752x480 @20 Hz, global shutter	1x IMU @200 Hz hard. sync.	Laser tracker @20Hz, VICON @100Hz, acc. $<$ 1mm	11 seqs, 0.9 km
Penn COSY VIO [96]	2017	Hand held	Indoor/ Outdoor	4x RGB 1920x1080 @30Hz. Stereo gray 752x480 @20 Hz. Fisheye gray 640x480 @30 Hz	2x accelero. @128 Hz, 2x gyros. @100 Hz, 1x acce./gyros. @200 Hz	Visual markers, acc. $<$ 15cm	4 seqs, 0.6 km
Zurich MAV [76]	2017	MAV	Outdoor	Mono RGB 1920x1080 @30 Hz, rolling shutter	1x IMU @10 Hz, 1x GPS, soft. sync.	Photogram. 3D reconstruction on Pix4D	1 seq, 2 km
Event-Camera[85]	2017	Hand held	Indoor/ Outdoor/ Synth.	Mono gray 240x180 @24 Hz, event camera	1x IMU @1kHz, soft. sync.	Motion capture system @200 Hz. high accuracy	27 seqs
TUM VI [109]	2018	Hand held	Indoor Outdoor	Stereo gray 1024x1024 @20 Hz	1x IMU @200 Hz, hard. sync.	Motion capture system @120Hz acc $<$ 1mm	28 seqs, 20 Km
UZH-FPV [27]	2019	MAV	Outdoor/ Indoor	Mono gray 346x260 @50 Hz, event camera. Stereo gray, 640x480 fish-eye	1x IMU, hard. sync.	Laser tracking system @20 Hz, acc. $<$ 1mm	27 seqs, 10 km

Table 4.1: A summary of the most common datasets for visual odometry.

However, to perform visual odometry, only the first eleven (00-10) sequences have ground-truth information; each sequence has a different frame length, as shown in Table 4.2. The available dataset to be used in training has then 23,201 frames in total. Each sequence's nature is also diverse: some sequences have a substantial variation in translation, others in rotation, and some sequences have both. Also, there is variation in the agent's speed. The images are grayscale for all sequences as exemplified by Figure 4.1. In this work, we only use images provided by the left camera.

	Sequence										
	00	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**
Frames	4,541	1,101	4,661	801	271	2,761	1,101	1,101	4,071	1,591	1,201

Table 4.2: The frame length of each sequence in the KITTI dataset [42].

Although the KITTI dataset was used to develop the final model, it imposes a greater difficulty to build new models from scratch due to a) the complexity of the input images (e.g., illumination variation, shadows, moving objects), and b) the requirement of a 6-DoF prediction. In this sense, we created two datasets with increasing degrees of difficulty. The first one, called the Pixel dataset, comprises 12,499 images of 100x100 pixels, in which the salient point is represented by several pixels in white color, and the background is

Figure 4.1: Sequential images from the KITTI dataset [42]. In this illustration, the images were picked up with an interval of 20 frames to highlight the differences among them.

filled with black color. The salient point is uniformly assigned and has a coordinate (x, y) concerning origin (0, 0) located on the image's top-left corner. An illustration of the Pixel dataset is shown in Figure 4.2. This dataset enabled us to perform regression between two salient points without paying attention to other visual information on the images.

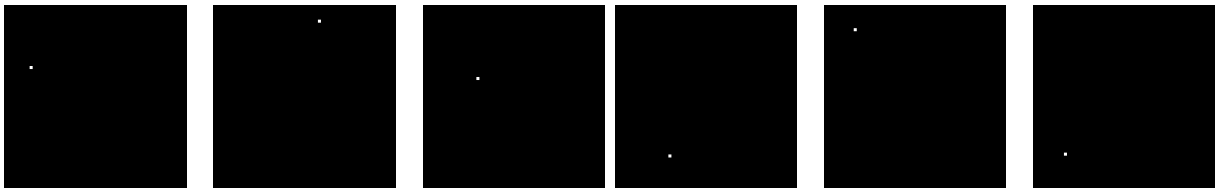


Figure 4.2: Sequential images from the Pixel dataset.

For more advanced experiments, we proposed the City dataset, which adds complex visual information. This dataset was created by linearly cropping 25,780 images from a high-resolution city landscape in portions of 300x300 pixels. The images were extracted in a continuous form, producing considerable overlap among them, which gives temporal information to the dataset, as shown in Figure 4.3. Similarly to the Pixel dataset, we saved the image's center coordinates (x, y) as ground-truth for each portion extracted. The City dataset allowed our architecture to understand complex spatial information in a temporal regression task.

Figure 4.3: Sequential images from the City dataset.

4.1.2 Metrics for Evaluation

The most common metrics used to evaluate the consistency compute the agent's absolute trajectory error (ATE) and the relative pose error (RPE). The dataset usually provides

the ground-truth values to compute these metrics. The ground-truth sequence and the predicted trajectory sequence must be time-synchronized, uniformly sampled, and have the same length.

Absolute Trajectory Error (ATE) is a metric primarily used to compute the method's global consistency. The estimated pose is compared with the ground-truth pose for each frame. However, the poses usually are specified in arbitrary coordinate frames and must be aligned to be compared. An aligning method should be used, such as the Umeyama method [124] or the Horn method [55], which finds a rigid-body transformation between the estimated and ground-truth pose.

The absolute pose error at instant i is given by

$$\mathcal{E}_i = \mathbf{G}_i^{-1}\mathbf{A}\mathbf{H}_i \tag{4.1}$$

where $\mathbf{G}_i$ is the ground-truth pose at instant i, $\mathbf{H}_i$ is the estimated trajectory pose at instant i, $\mathbf{A}$ is the best alignment transformation.

Traditional ways of computing the relative pose error for the entire trajectory is performed by the mean squared error (MSE) as

$$\mathrm{MSE}(\mathcal{E}_{1:n}) = \frac{1}{n}\sum_{i=1}^{n} \left\|\mathrm{proj}(\mathcal{E}_i)\right\|^2, \tag{4.2}$$

where n is the number of camera poses in the trajectory, and *proj* is usually the translation component [115].

An alternative way of computing it is by the root mean square error (RMSE), which amplifies the impact of outliers presented in the trajectory.

$$\mathrm{RMSE}(\mathcal{E}_{1:n}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \left\|\mathrm{proj}(\mathcal{E}_i)\right\|^2}. \tag{4.3}$$

Also, the median can be used to attenuate the influence of outlier even more than the MSE. In practice, ATE is a useful metric for visual inspection.

Relative Pose Error (RPE) was first introduced to calculate the error using only a relative relationship between the frames, which could solve the issues associated with a global frame comparison. In this sense, RPE measures the local consistency of the trajectory and is a reliable metric for the drift.

The relative pose error at instant i is given by

$$\mathcal{F}_i = \frac{\mathbf{H}_i^{-1}\mathbf{H}_{i+k}}{\mathbf{G}_i^{-1}\mathbf{G}_{i+k}}, \tag{4.4}$$

where $\mathbf{G}_i$ is the ground-truth pose at instant i, $\mathbf{H}_i$ is the estimated trajectory pose at instant i, and k is a fixed time interval. k determines the trajectory consistency accuracy, a small k intensifies the local drift, and a large k intensifies the global drift. For visual odometry, $k = 1$ is usually used because it provides the drift error frame-by-frame.

Similarly, RPE can be calculated for the entire trajectory using the mean squared

error (MSE) as

$$\mathrm{MSE}(\mathcal{F}_{1:n}) = \frac{1}{n-k} \sum_{i=1}^{n-k} \left\| \mathrm{proj}(\mathcal{F}_i) \right\|^2, \tag{4.5}$$

and the root mean square error (RMSE) as

$$\mathrm{RMSE}(\mathcal{F}_{1:n}) = \sqrt{\frac{1}{n-k} \sum_{i=1}^{n-k} \left\| \mathrm{proj}(\mathcal{F}_i) \right\|^2}, \tag{4.6}$$

where n is the number of camera poses and $proj(\mathcal{F}_i)$ is the projected component, which can be translational or rotational.

RPE can also be used to evaluate the global consistency by averaging over all periods. ATE and RPE are significantly correlated; however, RPE includes rotational and translation errors, while ATE commonly computes the translational error [115]. The KITTI dataset recommends evaluating the RPE by averaging all sub-sequences ranging from 100 to 800 meters.

4.1.3 Softwares, Libraries, and Tools

This work was implemented with the Python 3 programming language. To implement the neural networks, we preferred to use Pytorch due to its versatility to debug code and accelerate development.

In general, the main libraries used are the following:

- Matplolib: for data ploting;
- Numpy: for matrix calculations;
- Scipy: for scientific calculations;
- PIL: for image manipulation;
- Pandas: for tabular processing;
- PyTorch: for building neural networks;
- Scikit-Learn: for machine learning in general.

We also used the tool EVO [46] to align the odometry trajectory, compute the metrics and generate the statistics.

4.1.4 Hardware Specification

The hardware used for building and training the models has the following specifications:

- Motherboard: Asus Rog Strix Z490-E Gaming;
- CPU: Intel Core i7-10700KF @ 3.80GHz;

- RAM: Corsair DDR4 2x16 Gb @ 3600MHz;
- GPU: Nvidia RTX 2060 with 6Gb, and Cuda v11.1;
- Disk: Western Digital 1Tb;
- Operating System: ArchLinux v5.9.2.

4.2 Methodology

The methodology employed in this work will be divided into two parts, as follows:

1. Developing the architecture:
    - Extending the RAM architecture from classification to regression tasks. We will refer to this extended model as RAM-R. The Pixel dataset will be employed to train the model to regress the relative displacement between pixels. This step will generate a baseline model;
    - Increasing the problem's complexity by adding structured spatial information will require a better visual representation from the model. The City dataset will be used in this step, and the regressor must still produce the relative displacement between the images;
    - Improving the baseline model robustness by adding spatial structures (CNN) to the glimpse network, and temporal structures (LSTM) to the core network. We will refer to this extended model as RAM-RC. We will train this model on the City dataset;
    - Adapting the baseline model to the visual odometry context using the KITTI dataset [43] to train the model to regress the 6-DoF pose and learn geometry information in the glimpse network. This step will generate a visual odometry model. This model represents our RAM-VO architecture;
    - Replacing the REINFORCE [130] algorithm with the Proximal Policy Optimization (PPO) [111] to learn robust policies;
    - Employing the optical flow as contextual information to initialize the RL agent, and employing sequential information to represent the RL states better;
    - Performing studies on how the learned policies and hyper-parameters influence the architecture stability, solution quality, and computation cost.
2. Validating the architecture:
    - Employing the visual odometry model RAM-VO to generate trajectories from unseen sequences on the KITTI dataset;
    - Using the ATE and RPE metrics to evaluate the generated trajectories. Measuring the solution cost, quality, trade-off;
    - Discussing the results.

Chapter 5

RAM-VO

This chapter will present the RAM-VO construction, starting from its simpler version, RAM-R, a RAM extension to regression tasks, focusing ultimately on visual odometry. We will detail the overall implementations, such as increasing the network complexity, introducing spatial, temporal, and contextual structures, implementing the Proximal Policy Optimization (PPO), and configuring the glimpses hyperparameters and the RL rewards. Studies and discussions are presented to demonstrate our choices and results.

5.1 Extending RAM to Regression Tasks

The original RAM architecture was developed for simple classification tasks, as presented in Chapter 2. In this sense, we have developed a methodology to extend it to regression tasks, increasing the architecture's complexity to deal with more challenging input data. The methodology comprises two steps: a) adapting RAM to learn the displacement between single pixels in two different images; b) increasing the architecture complexity to provide the displacement between two images with complex visual structures. The next sections will detail each one of these implementations.

5.1.1 RAM-R: Simple Regression on Pixel Dataset

Adapting RAM to regression tasks to track image changes requires two glimpse networks, allowing the architecture to consume two different images simultaneously. Therefore, the modified RAM - from now on called RAM-R - can find the same features in both images and determine their correspondence. To do so, we changed the original decision network (our **Regressor network**) to generate linear outputs instead of class' probabilities. Besides, the architecture's capacity has been increased, which means the number of layers in each subnetwork increased, ensuring more representation power since the input information has doubled. Different from the original RAM, the policy's standard deviation is also learned during training, promoting exploration in the first epochs.

RAM-R is composed only of fully connected layers. The **core network** is a classic recurrent neural network; the layers use the rectified linear unit (ReLU) as activation functions, except the last layer in the **locator and regressor network**, which uses the hyperbolic tangent (tanh). We have not used any regularization techniques, such as

dropout and batch normalization. RAM-R architecture is shown in Figure 5.1; Table A.1 details the architecture parameters, number of neurons per layer, and activation function.

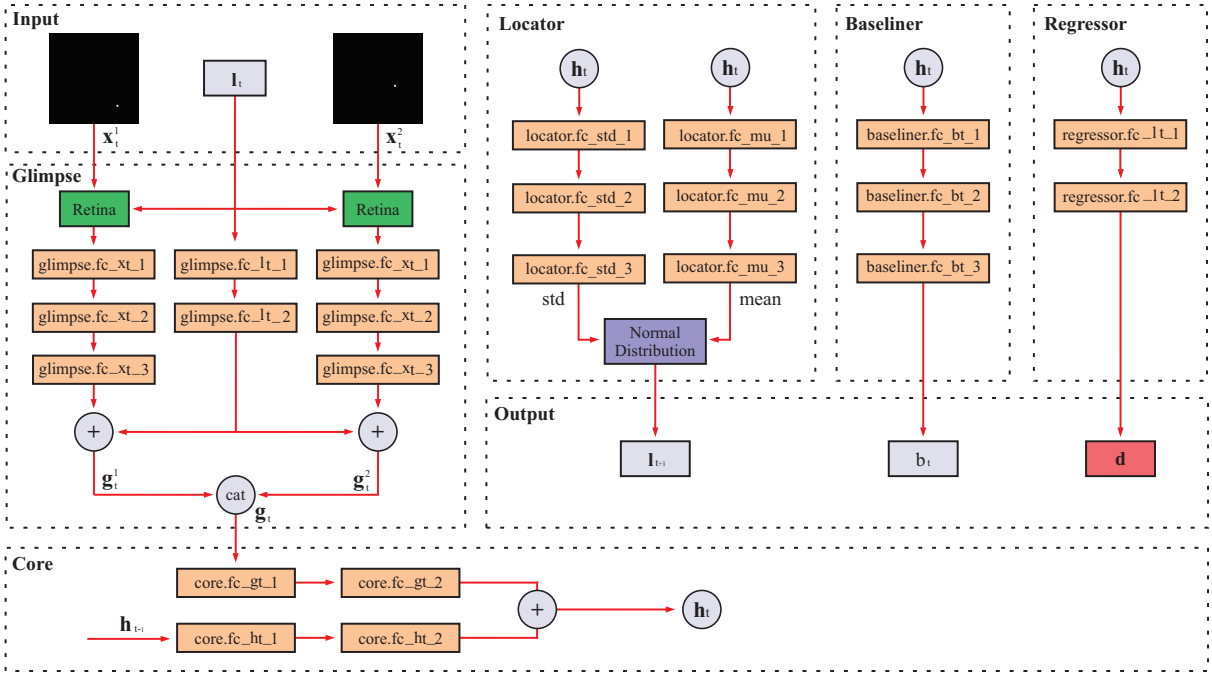


Figure 5.1: RAM-R: RAM architecture adapted to simple regression tasks. The modifications consisted of adding a second glimpse network, modifying the regressor network to allow linear outputs in the interval [-1, 1], learning the policy's standard deviation, and increasing the model's capacity.

RAM-R receives two images $(\mathbf{x}_t^1, \mathbf{x}_t^2)$ and a location of interest $\mathbf{l}_t$ as input. Each image is cropped with three increasing resolutions centered at the location provided to the retina sub-module. An average pooling is performed in each extracted patch to reduce its dimensions to a standard patch size defined as a hyperparameter. The three patches are flattened and concatenated into a single vector; then, they are processed through fully connected layers in the glimpse network; the location where the patches were extracted is added to the latent vector, originating a glimpse vector $\mathbf{g}_t^i$. We fuse the two glimpses by concatenating their latent spaces, producing the final vector $\mathbf{g}_t$. In this sense, the glimpse network captures patches in several resolutions from a determined location on both images, producing a latent vector combining the two information with their locations in the input.

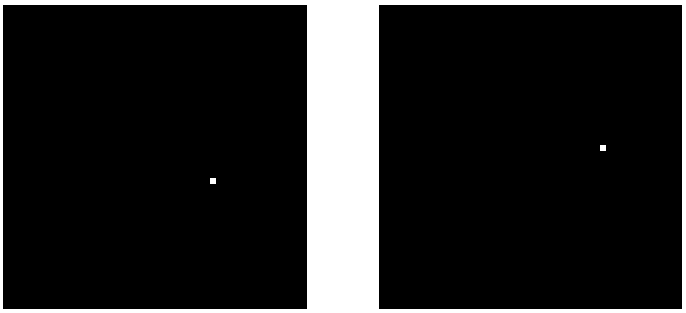
The **core network** integrates the received information $\mathbf{g}_t$ into its latent space $\mathbf{h}_t$, which is also composed of fully connected layers. This process is repeated several times according to the number of steps defined. For each step, the **locator network** reads the latent space $\mathbf{h}_t$, providing the next location $\mathbf{l}_{t+1}$. Only in the last iteration, when the core network generates the best latent space $\mathbf{h}_t$, the regressor network generate the prediction $\mathbf{d}$. The RAM-R architecture is trained via supervised and reinforcement learning. The locator network is detached from the backpropagation graph, which avoids gradients' propagation; therefore, the regressor, core, and glimpse networks are trained in a supervised learning fashion. The REINFORCE algorithm [130] is employed to train the locator network as in the original paper; however, we also learn the policy's standard deviation to promote

exploration in the first epochs. We jointly train a **baseliner network** to provide the state value b_t for each step and reduce the variance between the returns. More detail on the RAM's functionalities is provided in the next sections.

Several experiments were performed to reach this version. Some of them include: a) subtracting the latent spaces $\mathbf{g}_t^i$ in the glimpse networks; b) adding a second locator network, which allowed independent glimpse's location; c) increasing the number of glimpses; d) increasing the glimpse scale. However, modifications on the glimpse scale and quantity did not change the performance significantly due to the input data's simplicity. The addition of a second locator network slows down the learning process due to the necessity of learning two RL policies. The glimpses' subtraction causes the final vector $\mathbf{g}_t$ to keep unrepresentative information, decreasing performance. Therefore, we kept the architecture as functional and simple as possible. The complete architecture comprises 530,000 parameters.

Dataset creation and training stage. To validate the RAM-R architecture in simple visual regression tasks, we proposed the Pixel dataset. This dataset was created in several versions, with increasing image sizes. The first version consisted of images with 25x25 pixels where the white point measures 1 pixel. The second dataset was generated with images of 100x100 pixels; the white point consisted of 4x4 pixels. Furthermore, the third version consisted of images with 300x300 pixels. The results achieved were positive on all dataset versions. This section will report the results on the 100x100 version; the subsequent sections will deal with more complex image structures.

The input image pair consists of a black background and a white point in the foreground; the point location is randomly assigned, as shown in Figure 5.2. The purpose of the architecture is to determine the displacement (dx, dy) from one point to another. In this way, the dataset allows a maximum displacement of 100 pixels.



(a) Left glimpse sensor (b) Right glimpse sensor

Figure 5.2: Example of input for the left and right glimpse sensors. Each sensor receives a different image of 100x100 pixels. The white point consists of 4x4 pixels, and the goal is to determine the displacement between then.

Our experiments employed the z-score normalization on the training set, which guarantees the dataset has properties of a standard normal distribution. The ground-truth data were also standardized with the z-score function; this step is of high importance since large ground-truth values generate large gradients when compared to predictions, causing the problem of exploding gradient and making the model unstable – we want to be sure

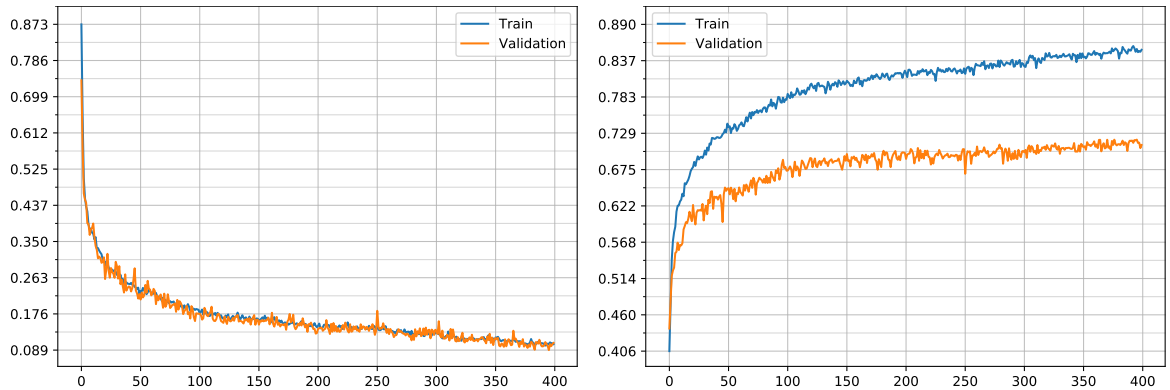
the gradients are inside a safe region. We define the reward function as the inverse MSE between the predictions and the ground-truth. We also train the supervised networks using MSE. We chose to capture five glimpses with patches of 8x8 pixels in resolution. Each glimpse is composed of 3 patches with resolutions multiple of 3 – we will detail them in the next section. The dataset separation is defined as 70% for training, 20% for validation, and 10% for testing. Table 5.1 resumes the hyperparameters used for training.

Glimpses	5
Patch Size	8
Patches	3
Patch Scale	3
Batch Size	128
Learning Rate Supervised	1×10^{-3}
Learning Rate RL	1×10^{-4}
Epochs	400
Training Samples	10,626
Validation Samples	1,249
Test Samples	624

Table 5.1: Hyperparameters employed for training RAM-R on the Pixel dataset.

We kept the learning rate for the reinforcement network at 1×10^{-4} due to the training's stability; high learning rate values tend to cause loss oscillation and make the model diverge. Also, larger batches help reduce the gradient variance in training, providing a better estimate for the update. We have not used early stopping or learning rate decay during training. The model was trained for 400 epochs and provided a good minimization, as seen in Figure 5.3.

The curves presented higher oscillation on the validation set due to the presence of fewer samples. The supervised loss presented a considerable minimization even for the



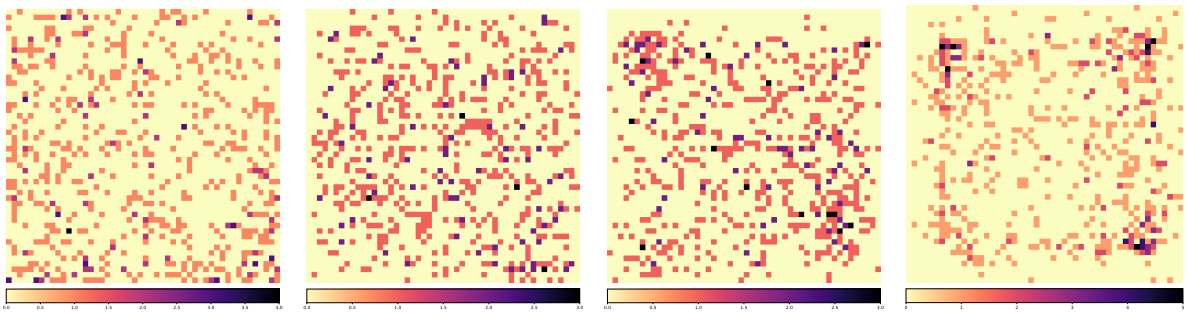
(a) Training and validation loss

(b) Training and validation reward

Figure 5.3: The evolution of training loss, validation loss, and reward. The validation supervised loss is close to the training loss, which indicates that the regressor is generalizing. However, the RL agent presents some difficulties in generalizing to unseen samples.

validation set; the reward signal has a discrepancy between the train and validation sets, which indicates a difficulty in generalizing to unseen samples. One possible explanation is the nature of the dataset, which presents zero values for most of the input image; the RL agent can understand that as being in the same state and continually reinforcing the same action. Although the location vector $\mathbf{l}_t$ is added to the current state, we noted a difficulty for the RL agent to make sense of it. In general, we considered that the model could reach a reasonable global solution without using regularization techniques.

Figure 5.4 presents a heat map with the glimpses locations distributed along with the training. We can observe that the glimpses are spread across the whole image at the beginning of training, indicating the agent has high entropy and consequently high exploration. As the training proceeds, the glimpses start to be gradually concentrated in regions around the image's corners. Indeed, this would be a good policy since capturing image patches from corner regions covers the whole image space, thus increasing the probability of finding the white pixel. We tested several policies to penalize glimpses outside the image space or increase the glimpse location's standard deviation; the policies learned slightly changed to select the same patches on the corners. However, we decided not to bias the RL agent towards a specific behavior for the final model. Therefore, we do not impose any restriction on the reward function besides MSE between the predictions and ground-truth.



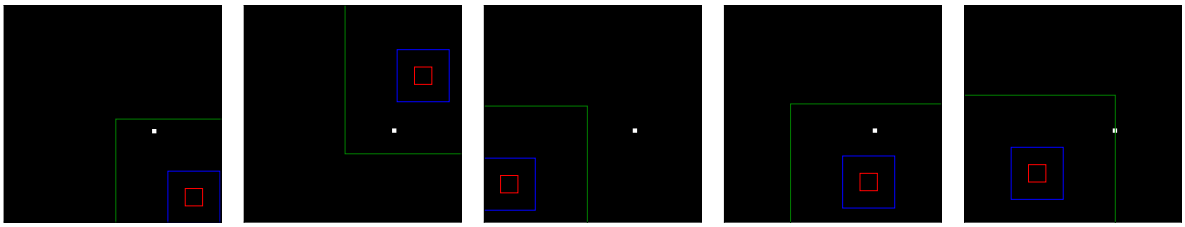
(a) Epoch 1 (b) Epoch 85 (c) Epoch 202 (d) Epoch 400

Figure 5.4: Heatmaps of glimpse's fixations during the training stage for a single mini batch. The glimpses are spread across the image on epoch one and gradually become concentrated on the corners.

Commonly, the weights of a neural network are kept at small ranges close to zero; however, for reinforcement algorithms, this hurts exploration. One way to solve this issue is to initialize the locator network's weights orthogonally - this guarantees a more diverse actions' selection in the first epochs. Regularization techniques such as dropout and batch normalization present some challenges for networks trained by reinforcement; it is vital that the state distribution be as stationary as possible during training; these techniques add significant noise to states and shift the mean to unknown regions. These alterations have a considerable impact on training and mainly in generalization; therefore, we prefer to avoid them.

Testing stage. The Pixel dataset was divided into a test set for testing our model on unseen data. The test dataset comprehends 624 image pairs and is normalized by the z-

score function with the same mean and standard deviation used for training; therefore, the test data is not biased. We can observe that the glimpses' locations match the fixations' heatmap generated by the learned policy, as shown in Figure 5.5.



(a) Glimpse 1 (b) Glimpse 2 (c) Glimpse 3 (d) Glimpse 4 (e) Glimpse 5

Figure 5.5: Glimpses' locations for the left sensor on the test set. Each colored square represents a glimpse scale. The five glimpses capture information from the pixel four times.

The first glimpse is performed in a random position; the agent defines the other four according to the policy. The colored squares correspond to the three input scales retrieved from the environment. Considering this experiment's simplicity, the RL agent can determine the displacement without using the high-resolution scale; only a raw representation is sufficient for a good prediction. However, we noted experimentally that just one pixel's capture is usually not enough for a reasonable prediction; therefore, using a small patch scales hurts performance. One way around is to use large scales, which allow more than one capture. For this test sample, the totality of five glimpses captures the pixel four times.

The first predictions in the test set are shown in Table 5.2. The Mean Absolute Error (MAE) was employed to compare the results because the pixel's displacement respects the Manhattan's distance. Therefore, MAE provides the distance between the pixels, facilitating the comprehension of the results. The final results demonstrate that the RAM-R could infer the pixel's displacement between the two images with small average

Prediction (dx, dy)	**Ground-truth (dx, dy)**	**MAE**
-0.98, -53.19	-1.70, -54.70	2.230
-2.85, -13.37	-5.70, -13.70	3.180
34.75, -22.5	38.30, -23.70	4.750
33.81, 43.45	32.30, 41.30	3.660
50.89, 31.81	58.30, 29.30	9.920
-28.22, 31.07	-28.70, 29.30	2.250
4.38, -41.38	1.30, -36.70	7.760
60.07, -6.21	64.30, -8.70	6.720
82.72, -2.84	81.30, -2.70	1.560
20.94, 2.86	22.30, 1.30	2.920
	Mean	**4.495**

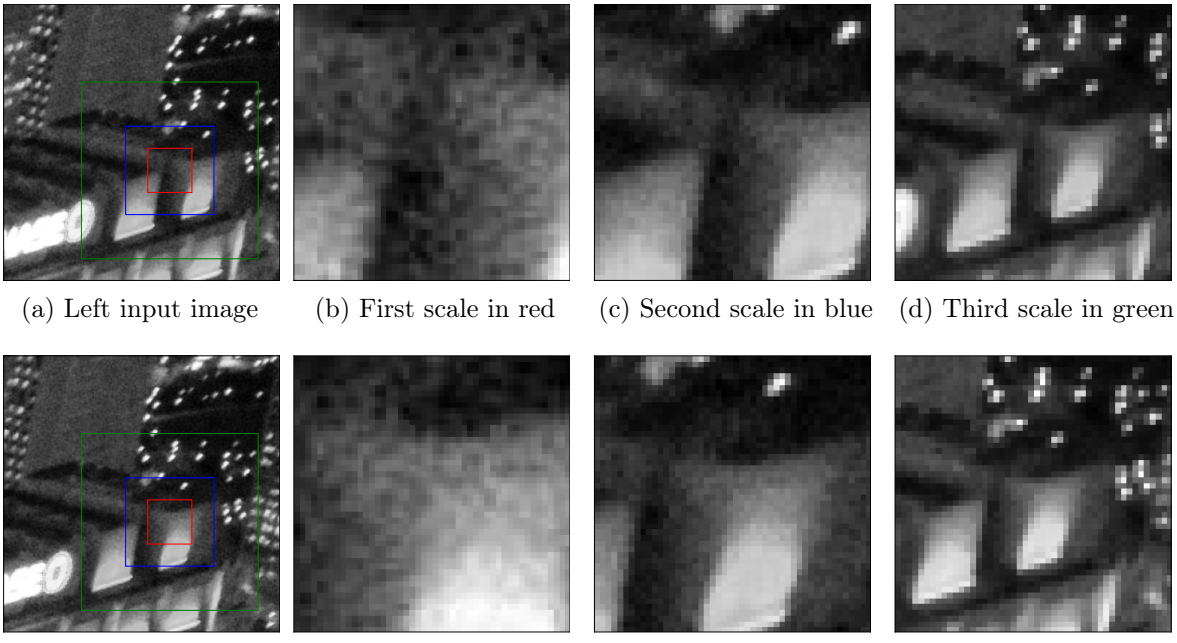
Table 5.2: The first results reported for the test set on the Pixel dataset. The displacement is given for $(x,\ y)$ coordinates. The average MAE is considered acceptable.

errors. For the complete test set, the final MAE was 3.118, which corresponds to an error of 3.19% for images of 100x100 pixels. We also had similar results for input images of 300x300 pixels. Therefore, we understand that the predictions were good enough to continue increasing the input image's complexity. In this sense, we conclude that RAM can be extended to perform regression tasks, confirming the hypothesis H_1.

5.1.2 RAM-RC: Complex Regression on City Dataset

Although the Pixel experiments' results are acceptable, the problem is straightforward, and the glimpse scales are not fully useful. The integration of information in the core network is also problematic due to null inputs most of the time – the RL agent has difficulties differentiating between states. To make the regression more complicated, we propose the City dataset, which comprises images with 300x300 pixels extracted from a city's panoramic image in high resolution. This dataset provides an important environment to understand the agent's behavior when dealing with highly non-linear visual information. In this sense, we can observe how the agent uses the glimpse scales to choose an action; and how the sequence of observations generates a usable latent space for predictions.

A significant feature presented in RAM's formulation is the image scales, which simulate the human visual system. The first scale corresponds to a high-focus region with a smaller dimension at the center; the other scales have large dimensions but lower resolutions, as exemplified in Figure 5.6. The designer can define the number of scales; we prefer to keep the model with three scales, considering the input image's size.



(a) Left input image (b) First scale in red (c) Second scale in blue (d) Third scale in green

(e) Right input image (f) First scale in red (g) Second scale in blue (h) Third scale in green

Figure 5.6: Example of an input image pair. Each glimpse sensor extracts three patches with increasing scale size; then, they are resized to a standard patch size with the same size as the first scale. In this example, the first scale is set to 48x48 pixels; the others are two times larger than the previous.

This pyramidal-like structure provides a trade-off between the amount of information and computational cost; that is, the agent can observe the environment's details in high-resolution with the first scale; and observe the most salient elements presented on the boundaries with the others. The second and third scales are resized to the same size as the first one using average pooling. Thus, the image patch from the first scale does not undergo any modifications, whereas the other two patches are changed by the average of the pixel values in its proximity. In practice, the network deals with three input images with the same shape for each glimpse sensor.

Conceptually, the agent will use this peripheral information to determine the next focus location $\mathbf{l}_{t+1}$ for the next glimpse. For this purpose, the glimpse network was modified to interpret structured visual elements through convolutional networks (CNN). Thus, three independent convolutional layers process each glimpse scale without any pooling layer between the CNNs. We found it reasonable to use a kernel size of 5 and to generate four channels for the first CNN layer and eight channels for the second. More channels should improve the model even further; however, pooling operations or increasing the stride become necessary to avoid increasing parameters when flattening the resulting vector.

Figure 5.7 exemplifies the RAM-RC model used in this experiment. It includes more sophisticated memory elements such as the Long Short-term Memory (LSTM) in the **core network**, allowing the model to track long-distance dependencies and better represent the predictions. LSTM structures diminish the problem of vanishing gradients during training, stabilizing the model. We propose only one LSTM layer for this problem although more layers allow the model to have a hierarchical representation of the sequential data. More details on model configurations are depicted in Table A.2.

The addition of convolutional layers added a relevant cost to the model. Although CNN has shared parameters, the cost of convolution operations makes training more

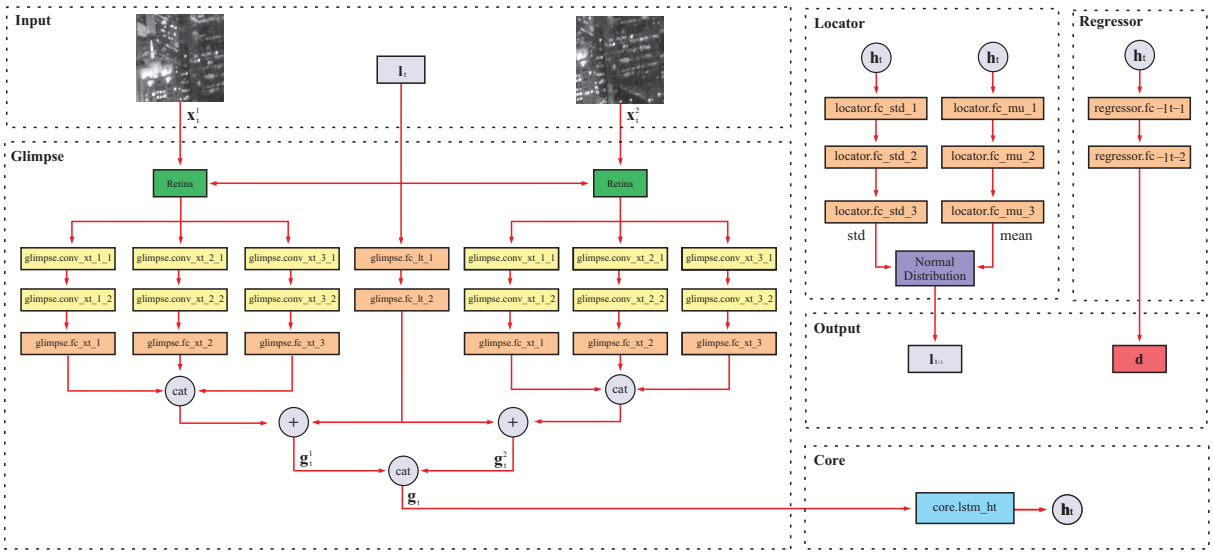


Figure 5.7: The RAM-RC architecture adapted to complex regression tasks. The modifications consisted of adding CNN layers on the glimpse network and an LSTM layer on the core network. These elements provide a better spatial and temporal representation's capacity to the model. The baseliner network is omitted.

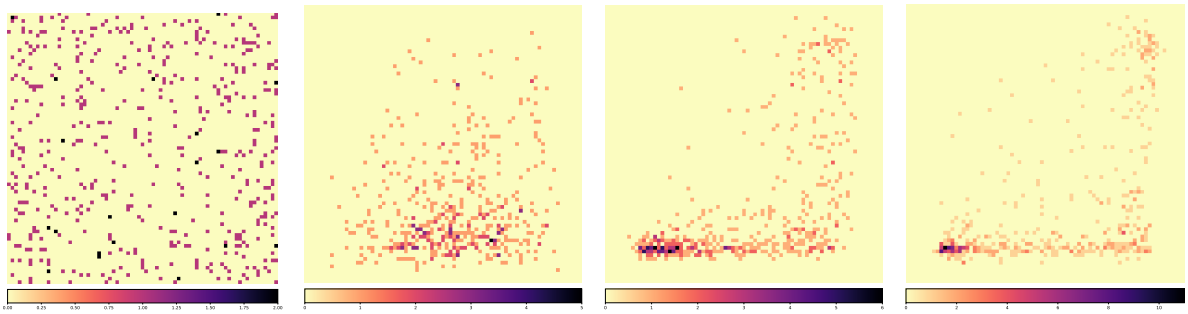
expensive. Also, the use of LSTM significantly increases the number of parameters in the model. We counterbalanced this issue with the use of fewer channels and smaller patches. LSTMs were also maintained with the smallest hidden space that would provide acceptable results. In total, the model has 2 million parameters.

Training stage. Like the Pixel experiment, the goal here is to predict the displacement (dx, dy) between two complex input images by iteratively building a latent space $\mathbf{h}_t$. Most of the experiment configuration stays the same, except for patch size, which has been increased to 32x32 pixels. This change improves the information flow since the image was increased from 100x100 to 300x300 pixels; therefore, the major scale represents around 10% of the original image. We also decreased the learning rate to maintain the model's stability, and the epochs were increased to 700. Table 5.3 resumes the hyperparameters used for training.

Glimpses	5
Patch Size	32
Patches	3
Glimpse Scale	2
Batch Size	128
Learning Rate Supervised	1×10^{-4}
Learning Rate RL	1×10^{-5}
Epochs	700
Training Samples	21,897
Validation Samples	2,575
Test Samples	1,287

Table 5.3: Hyperparameters employed for training the RAM on the City dataset.

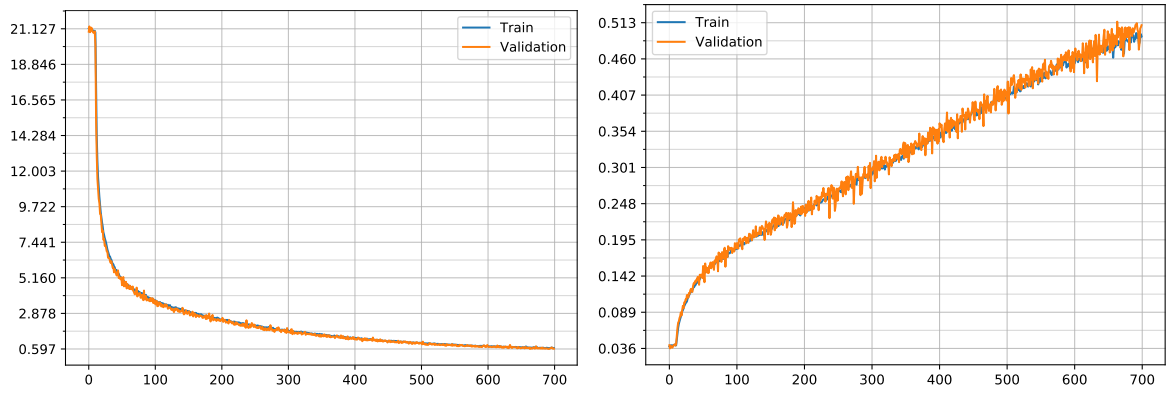
During training, we can note that the agent starts with much exploration in epoch one and gradually makes fixations horizontally and vertically in the image, as shown in Figure 5.8. This behavior is possibly due to the City dataset's dynamics, which varies linearly on the axes x and y; thus, the policy's refinement during training reveals that the agent is learning the most natural way of selecting the appropriate data to the supervised network. In the end, the fixations tend to be primarily concentrated in a small region at the bottom. For this experiment, only a small portion of the image is sufficient to make a good prediction; fixations in distant portions do not provide more information for the problem's solution; therefore, the learned policy aims to capture and exploit a small region without concern about the rest of the image. This kind of regression is commonly solved using larger convolutional networks that capture the entire image. However, consuming the whole image makes the architecture more computationally expensive and not a scalable solution for higher resolution images, such as those used in visual odometry. Therefore, using the innovative concept of attention guided by reinforcement learning allows the model to determine what is important to consume for the problem's solution – some policies can be very counterintuitive for the designer.



(a) Epoch 1 (b) Epoch 200 (c) Epoch 500 (d) Epoch 700

Figure 5.8: Heatmaps of glimpse's fixations during the training stage for a single mini batch. The glimpses are spread across the image on epoch one and gradually become concentrated on a single region.

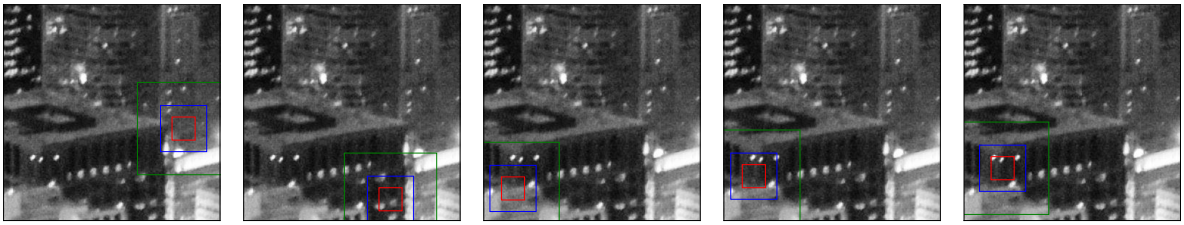
Like the previous experiment, the supervised validation loss presented a behavior similar to the training loss, which indicates that the model can learn information that allows generalization for unseen data. This time, the reward curve indicates that the RL agent can make good predictions for the validation set; the observed difference when compared to the Pixel experiment is due to the higher image variability presented on the City dataset, which allows the model to build a robust state representation. Figure 5.9 presents the training and validation losses, as well as the reward achieved by the agent.



(a) Training and validation loss (b) Training and validation reward

Figure 5.9: The evolution of training loss, validation loss, and reward. The validation loss is close to the training loss, which indicates that the model is generalizing for unseen samples. The reward curve indicates the RL agent can provide a good estimate in the validation set.

Testing stage. RAM-RC was tested on data unseen during training. We can observe that the glimpses' behavior is much more concentrated on a single region than scanning the whole image, as seen in Figure 5.10. The scales play a role after the second glimpse when the agent explores regions of a high gradient in the image, facilitating the displacement's identification.



(a) Glimpse 1 (b) Glimpse 2 (c) Glimpse 3 (d) Glimpse 4 (e) Glimpse 5

Figure 5.10: Glimpses' location for the left sensor for test set on City dataset. Each colored square represents a glimpse scale.

The first predictions in the test set are shown in Table 5.4. Similarly, the MAE was used to determine the displacement between the two images and evaluate the results' quality. An average MAE of 2.118 is considered acceptable for the displacement estimation in the City dataset – for the whole test set, MAE was 3.810.

Prediction (dx, dy)	**Ground-truth (dx, dy)**	**MAE**
1.64, 21.36	1.00, 23.00	2.280
7.32, -16.67	9.00, -16.00	2.350
-3.95, -1.35	-4.00, 0.00	1.400
3.20, -29.82	5.00, -33.00	4.980
-30.46, -0.24	-33.00, 0.00	2.780
17.45, 3.00	17.00, 3.00	0.450
-14.81, -1.19	-15.00, 0.00	1.380
11.83, 40.72	9.00, 42.00	4.110
49.10, 23.08	49.00, 24.00	1.020
-6.82, -0.25	-7.00, 0.00	0.430
	Mean	**2.118**

Table 5.4: The first results reported for the test set on City dataset. The displacement is given for $(x,\ y)$ coordinates.

Compared to the Pixel's experiments, the results on City present a lower error; we attribute this to the use of more sophisticated spatial and temporal elements. The use of a CNN allowed the glimpse network to identify the most interesting spatial regions at different scales and target them in the following glimpses. Also, LSTM layers in the core network offered a better capacity to integrate the information from the five glimpses and produce a representative latent space for regression. The experiment with complex visual information allows the RL agent to determine a robust policy; from the results, we conclude that just a small region is sufficient for good quality predictions. Therefore, we confirm the hypothesis H_2 in which RAM-RC can be used for regression tasks with complex visual data.

5.2 RAM-VO: Visual Odometry Regression

Visual odometry regression is substantially more complex than the experiments reported so far. The 6-DoF regression makes the predictions trickier; the input images have a higher resolution, requiring a better state representation. Therefore, this section will detail the modifications made to creating the RAM-VO, such as increasing the architecture's capacity, adding contextual information, and implementing the Proximal Policy Optimization (PPO) [111] algorithm. We will present a baseline version and incrementally add the other functionalities. The architecture presented here corresponds to the final version; however, several experiments have been carried out to reach this point; we will discuss the relevant modifications along the text.

Initially, the RAM-RC was adapted to allow 6-DoF regressions; thus, the network regressor's capacity doubled to allow the rotational and translational predictions independently. The glimpse network has also been changed to promote the learning of the scene's geometry. In the previous experiments, the RAM-RC used independent convolutional filters on each input image and delegated the integration of the whole latent spaces $\mathbf{g}_t^i$ to the core network; which is inefficient because it promotes the learning of the image's appearance, and, in visual odometry, we are interested in learning the geometric relations and their correspondence between frames. To solve this issue, we were inspired by the FlowNetS [36] architecture, in which the input images are concatenated and used into the convolutional channels; this alternative design favors the model to capture the scene's geometry, especially the optical flow. In this sense, RAM-VO still has three convolutional pipelines but processes the two images simultaneously. The proposed changes are shown in Figures 5.11 and 5.12. For more information on the model's configurations, see Table A.3.

Learning the image's appearance significantly overfits the network; that is, the predictions for the seen sequences present low errors, whereas the model cannot generalize with

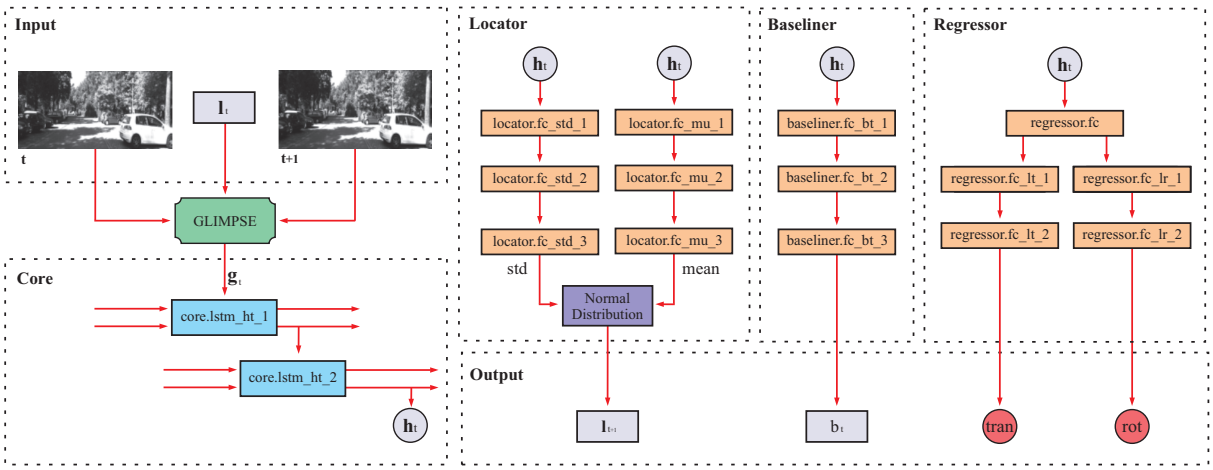


Figure 5.11: The RAM-VO architecture for the baseline version. The modifications included increasing the glimpse network capacity, providing the image patches into convolutional channels, adding two LSTM layers in the core network, and doubling the regressor network's capacity to enable 6-DoF pose regression. Figure 5.12 shows the Glimpse.

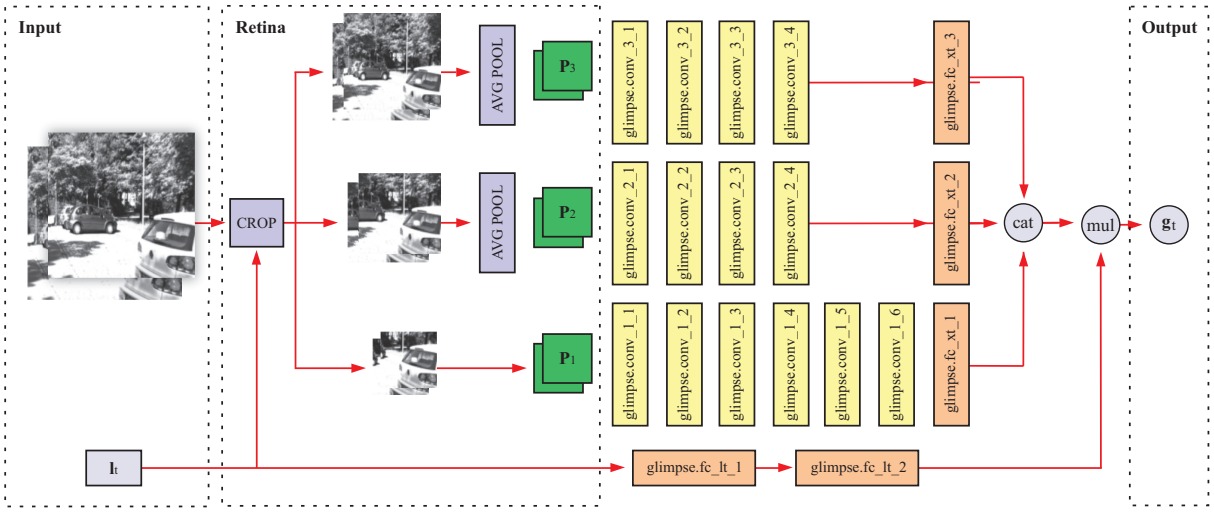


Figure 5.12: The glimpse network configuration. The input images are cropped at the location $\mathbf{l}_t$, generating three patches $\mathbf{P}_i$ for each image. Then, the patches are concatenated by their scales and processed by independent CNNs; the resulting latent space is multiplied by the location $\mathbf{l}_t$, providing the final vector $\mathbf{g}_t$.

quality to unseen sequences. We understand that the ability to generalize well is closely associated with learning the scene's geometry; learning appearance does not provide good generalization results in our case. Therefore, RAM-VO learns the optical flow in the glimpse network, already integrating the two image's information and releasing the core network to deal only with the integration of observations – these structural modifications remove inefficiencies like redundancy and bottlenecks.

Considering that the convolutional operations are performed on both image patches simultaneously, the application of the same filters considerably reduces the computational cost compared to performing the features' extraction separately. However, convolutional operations still add a high cost to the model; although the number of parameters has decreased due to their sharing, the cost of applying the filters increased. We also tested the viability of a fully connected glimpse, but losing spatial relations decreases the performance. Therefore, we maintained the convolution operations but use only 6 layers for the high-resolution scale, with 128 channels in the last operation. In contrast, the DeepVO [126] architecture employs 9 CNN layers with up to 1,024 channels in the last layer.

In this sense, we observed that convolutional operations with smaller kernels significantly minimize the loss – which is expected since smaller kernels enable the detection of finer features. Therefore, we chose to keep the high-resolution scale with a kernel size of 3x3 pixels and the other scales with a 5x5 kernel since they represent large areas. Padding values are defined as zero to avoid the reduction in the input size across the CNN layers. Based on the FlowNetS [36] model, pooling operations are entirely removed, and the input information's dimensionality is reduced by varying the stride between 1 and 2 during the filters' application. Pooling operations have the premise of reducing the information amount that flows through the network by calculating either the average or the maximum value of a group of neurons. However, this kind of information reduction impacts regres-

sion in visual odometry, mainly the translational component. From a visual inspection of the input images, we can observe that the translation is the component that varies least from one frame to another. In this way, the glimpse network captures the same image region with only subtle differences between frames; when the pooling operation is used, this difference becomes imperceptible.

The supervised loss and the reward function are both defined in terms of the MSE. MAE was also tested as a reward function; however, it does not penalize outliers; and in visual odometry, we want to minimize the outliers as much as possible since only one poor prediction can harm the entire trajectory. Therefore, the supervised loss L is defined as

$$L = \frac{1}{N}\sum_{i=1}^{N} \|\hat{\boldsymbol{p}} - \boldsymbol{p}\|_2^2 + k\,\|\hat{\boldsymbol{\varphi}} - \boldsymbol{\varphi}\|_2^2\,, \tag{5.1}$$

where $\hat{\boldsymbol{p}}$ and $\hat{\boldsymbol{\varphi}}$ are the position and orientation prediction, respectively; $\boldsymbol{p}$ and $\boldsymbol{\varphi}$ are the ground-truth values; and k is the constant factor weighting the two losses; we defined it as 1. The reward function R is then defined as

$$R = \frac{1}{1+L}. \tag{5.2}$$

Similar to previous experiments, we prefer not to bias the RL agent towards a specific behavior; therefore, only the visual odometry error is employed in the reward function. The weighting constant k can also be altered to favor one component over another; this is specially done when the ground-truth values are not normalized, and the orientation component must be compensated since they present a lower variation range. Tests were carried out without normalization to determine the best value for k; we tested values of 50, 100, and 200. The test's results indicate that the trajectory's prediction becomes more accurate for some sequences and worse for others. Therefore, we preferred to maintain the ground-truth values normalized and $k = 1$ for this baseline version. Also, we composed the orientation component $\boldsymbol{\varphi}$ with the Euler angles as roll ϕ, pitch θ, and yaw ψ; the position $\boldsymbol{p}$ is composed of the coordinates x, y, and z. In conclusion, the RAM-VO's goal is to regress the 6-DoF vector $[\phi, \theta, \psi, x, y, z]^T$.

Data analysis and pre-processing. Before training, we analyzed the motion dynamics present in the KITTI dataset sequences. There is a considerable average variation in motion for rotational and translational components between sequences, as shown in Table 5.5. Sequence 1 has a higher average variation for the translational component z concerning the other sequences; furthermore, sequence 1 has fewer samples, compromising learning. Sequence 0 is the one with the highest average variation concerning the rotational motion. We observed that the architecture could not generalize with quality for the sequences with little data and significant differences in the motion average. This fact was expected since the architecture would have a hard time generalizing such disparity to training data. Therefore, we chose to carry out the training in long sequences and those with a regular variation in the average motion, leaving the others sequences to validate and test. In this sense, sequences 0, 2, 4, 5, 6, 8, 9 were used for training, sequences 10 for validation, and sequences 3, 7 for testing. We did not use sequence 1. We also used this split to compare the results with other methods.

Sequence	$\bar{\phi}$	$\bar{\theta}$	$\bar{\psi}$	$\bar{x}$	$\bar{y}$	$\bar{z}$
0	**0.8316**	**0.6113**	**0.8194**	0.6243	0.4985	0.6384
1	0.3605	0.3742	0.3839	0.5601	1.1367	**2.1863**
2	0.7071	0.5266	0.7476	0.5814	0.5602	0.5751
3	0.6508	0.3233	0.7225	0.3229	**1.1602**	0.6588
4	0.3226	0.0353	0.5380	0.1962	0.6007	0.8007
5	0.5838	0.4234	0.5677	0.4214	0.4137	0.6601
6	0.4506	0.3841	0.3633	0.3838	0.4686	0.6610
7	0.5545	0.5906	0.5811	0.5725	0.5269	0.8050
8	0.6392	0.5121	0.6957	**0.6336**	0.7961	0.6666
9	0.5834	0.6021	0.6767	0.6244	0.4729	0.6104
10	0.7741	0.4521	0.7678	0.4786	0.5189	0.7259

Table 5.5: Motion component's mean variation for each sequence in the KITTI dataset. Each component is standardized independently. Some sequences present a higher mean variation, which impacts the model's generalization capability.

The monocular input images are captured in real-world environments with 1200x360 pixels in resolution. Initially, the pixel intensity histogram was equalized by the Contrast Limited Adaptive Histogram Equalization (CLAHE) method, which equalizes the histogram in small windows of 8x8 pixels, preventing noise from a small part be extended to the entire image. In this way, we can highlight the image's features without increasing the noise, as seen in Figure 5.13. The histogram equalization was especially useful to provide smoother optical flow maps, as presented in the next section. The training images were then normalized with the z-score function before entering the glimpse network due to the sequences' significant illumination variation.

(a) Before equalization (b) Before equalization

(c) After equalization (d) After equalization

Figure 5.13: The impact of adaptive histogram equalization for samples from sequences 7 and 8. The features are highlighted without increasing noise.

When performing experiments with higher RL learning rates than those chosen, we observed RAM-VO concentrates most of the glimpses in the image's borders. In those cases, for which the REINFORCE algorithm diverged, we consider this behavior a su-

pervised convergence to a local minimum – hence, the generalization capacity is strongly affected in those situations. Although the RL agent diverges, the tendency to look at the image's corners can also be seen as an attempt to cover regions with a more meaningful data variation for the problem. The optical flow on the image's border presents the highest magnitude, mainly for translational motions, as shown in Figure 5.14. Therefore, the RL agent can be precipitately attracted to these regions and end up learning a low-quality policy that always favors data input from the borders without exploring alternative sequences of observations.

(a) Optical flow during a rotational motion.

(b) Optical flow during a translational motion.

Figure 5.14: Sparse optical flow extracted for rotational and translational motion for sequence 0 in the KITTI dataset. For a rotational motion, the optical flow is concentrated mainly in the image's center; for a translational motion, it is distributed around the image's border.

Understanding how the optical flow behaves helps us understand whether the learned policy can favor the supervised network to generalize. Ideally, the RL agent should capture data from regions that provide both rotational and translational information. Once the proportionality constant $k = 1$, no preference is given for one component over the other in the baseline version. We can note that the image's center does not provide meaningful information for translation from the sparse optical flow, but it does provide for rotation. In this sense, a complementary behavior seems to exist between rotation and translation.

Training stage. Despite the RAM-VO separation into several distinct sub-networks, the model can be understood only as two separate networks; the first is trained by supervised learning and the second one by reinforcement. The **locator network** controls the input information so that the supervised network can generate predictions. In this sense, the correct alignment of the two networks is very delicate – a rapid convergence of the

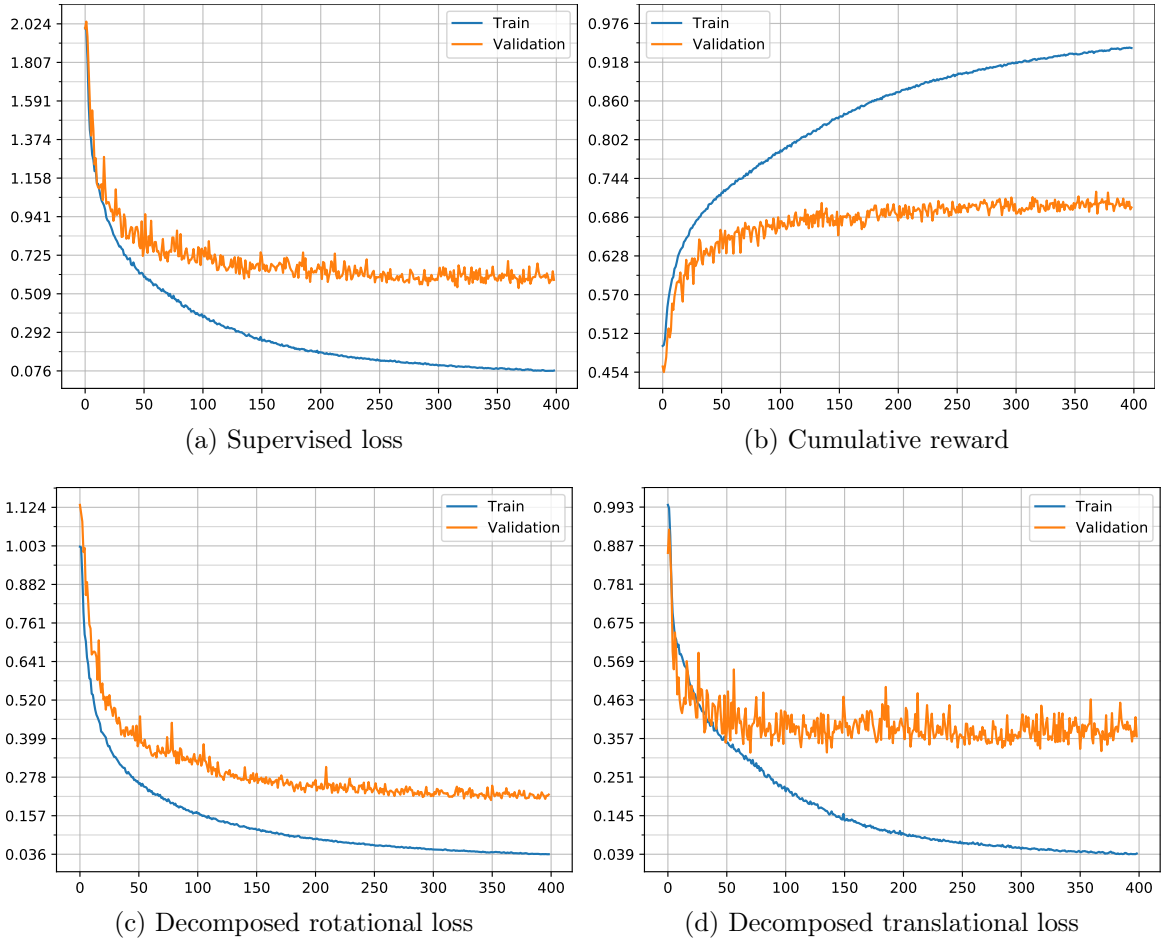
supervised network generates an ineffective policy, which tends to have the same behavior regardless of the input images. In this case, the model arrived at a suboptimal solution, which inhibits more exploration and stuck the model at local minima. On the other hand, a supervised network that does not learn makes the reinforcement network unstable since it depends on the reward function. Adjusting the hyperparameters and the reward function adequately to allow the two networks' convergence was the most complex task we handle. For example, the definition of an adequate learning rate significantly impacts the supervised network's convergence speed and, consequently, the reinforcement network's stability. In general, we kept the learning rate for the reinforcement network lower than the supervised one. Also, the batch size plays an important role when training by reinforcement. The return has a significant variance across the episodes; this dramatically slows the learning process. One alternative is to increase the batch size; however, this can be done only to a limited degree, usually defined by the GPU's memory. Table 5.6 resumes the hyperparameters used in training.

Glimpses	8
Patch Size	32
Patches	3
Glimpse Scale	3
Batch Size	128
Learning Rate Supervised	1×10^{-4}
Learning Rate RL	1×10^{-6}
Epochs	400
Training Samples	18,990
Validation Samples	1,200
Test Samples	1,902

Table 5.6: Hyperparameters employed for training the baseline RAM-VO on the KITTI dataset.

For the experiments, we varied the number of glimpses from $1-8$ with a patch size of 32x32 pixels and three scales; for all cases, the totality of input information corresponds to only a small percentage of the input (5.7% of the total available for 8 glimpses) – this is far less compared to methods that deal with the whole image. The whole train set comprehends 18,990 image pairs; although the training is possible with this amount, more data should significantly improve the results – training in complex environments requires as much data as possible to cover the many dynamics presented (e.g., people walking, cars, lens distortion).

The training was performed for 400 epochs and presented a minimization of the supervised training loss and maximized the agent's cumulative reward. However, the validation loss has not been minimized to the same extent; more training epochs would not solve the issue since it would overfit the model even more. Still, this generalization capability is superior to that presented by a RAM-VO version that learned the image's appearance, for instance. A better understanding can be achieved by decomposing the supervised loss into its rotational and translational components, as shown in Figure 5.15.



(a) Supervised loss

(b) Cumulative reward

(c) Decomposed rotational loss

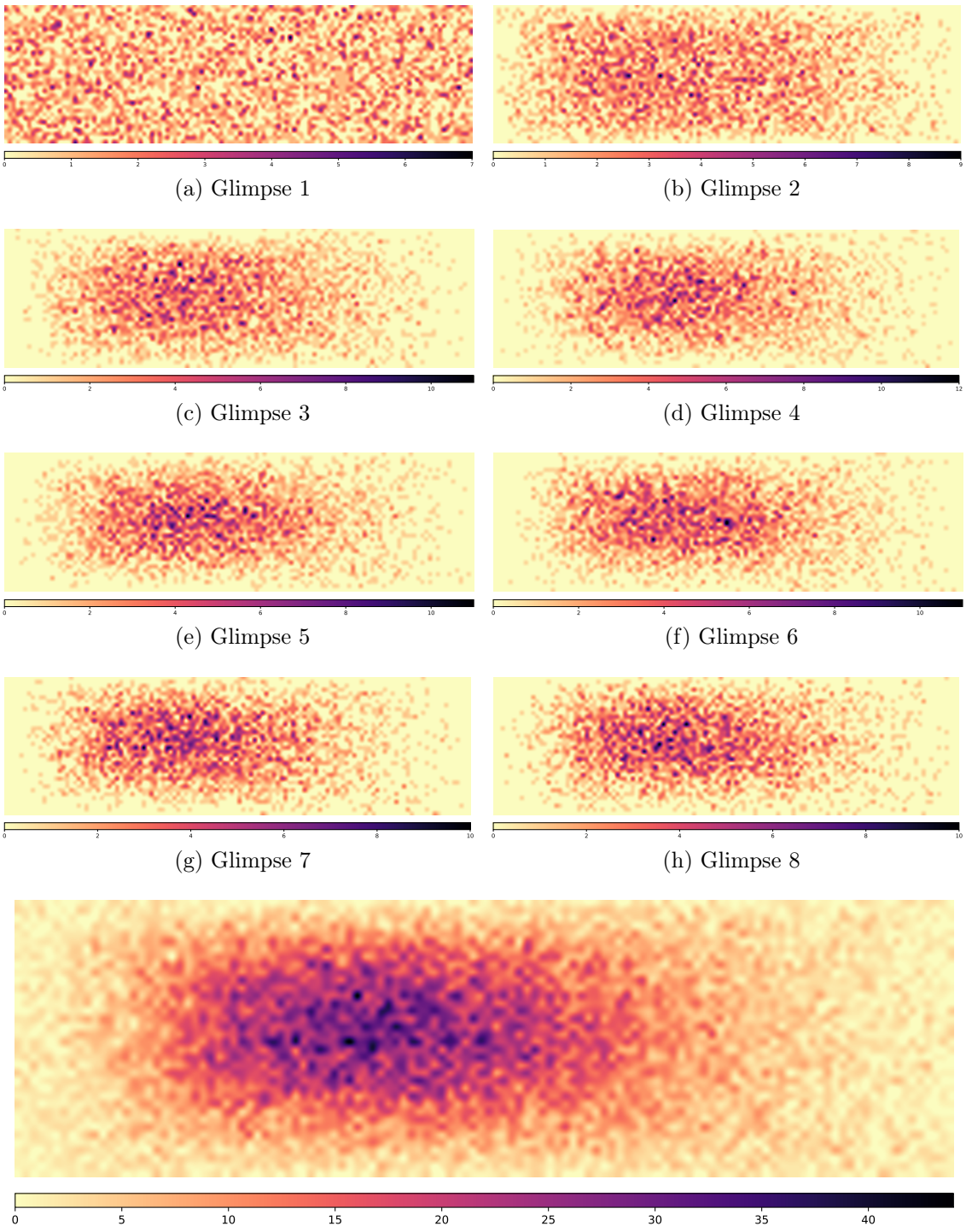
(d) Decomposed translational loss

Figure 5.15: The evolution of training loss, validation loss, and reward for the baseline RAM-VO. Also, the supervised loss is decomposed into its rotational and translational components. The baseline model presents a difficulty to minimize the validation losses, especially the translation one.

Although the supervised network can minimize the rotational loss to a certain extent, the translational loss minimization presents some difficulties. This possibly occurs due to the ambiguity in determining depth from monocular images. Methods based on unsupervised learning like GANVO [1] learn to predict the depth map, which helps determine the correct 3D position in space. A second alternative is to integrate inertial information such as IMUs. It is important to note that learning the scene's geometry had significantly diminished both the translational and rotational errors in the test set; the apparent rotational motion presents higher variation between frames when the car is rotating, contributing to its minimization, while the apparent translational motion is subtle and predominant on the image's borders.

From the heatmaps of glimpses' fixations on Figure 5.16, we can see that the majority is concentrated between the image's center and the left border. This contributes to a more significant minimization of the rotational loss, although the translation is also contemplated. The first glimpse is randomly defined, and the RL agent chooses the remaining. Lowering the RL learning rate contributes to stabilizing the agent, which promotes greater observation diversity. We can also observe that the entire image is not necessary to pre-

dict motion; similar to the City experiment, only a restrained region is sufficient. From the composed heatmap (i), we cannot distinguish between individuals glimpses, although the typical Gaussian pattern for stochastic policies is well observed. Figure B.1 contains the trajectories learned during training.



(a) Glimpse 1 (b) Glimpse 2

(c) Glimpse 3 (d) Glimpse 4

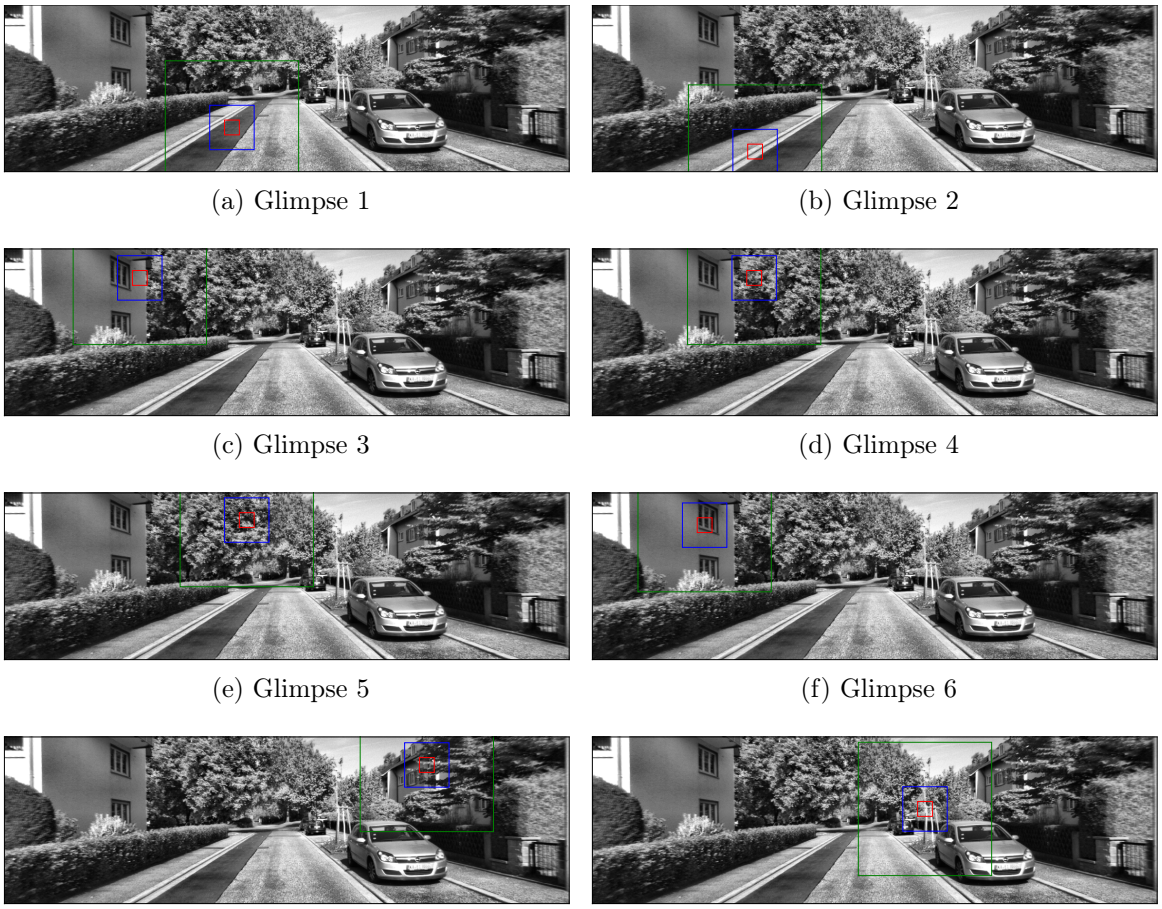
(e) Glimpse 5 (f) Glimpse 6

(g) Glimpse 7 (h) Glimpse 8

(i) All glimpses

Figure 5.16: Heatmaps of glimpses' fixations for the entire sequence 2. The first glimpse is randomly defined, and the RL agent chooses the other 7. Most of them are concentrated between the image's center and left border.

The observations sequence depicted in Figure 5.17 shows that the glimpses are located according to the heatmap, and the tendency is to explore high-gradient regions, which are more informative for the problem. The glimpse sensor seems to use the peripheral information only to some extent; one possibility is that the RL agent is learning a complementary behavior. The scales information can be used to discard the current region and jump to different ones in the expectation of finding more informative features.



(a) Glimpse 1 (b) Glimpse 2

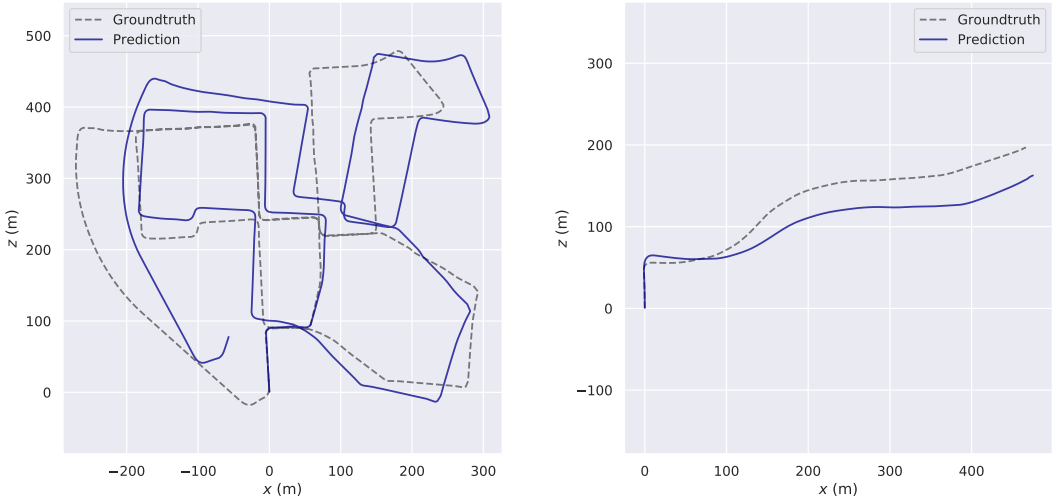
(c) Glimpse 3 (d) Glimpse 4

(e) Glimpse 5 (f) Glimpse 6

(g) Glimpse 7 (h) Glimpse 8

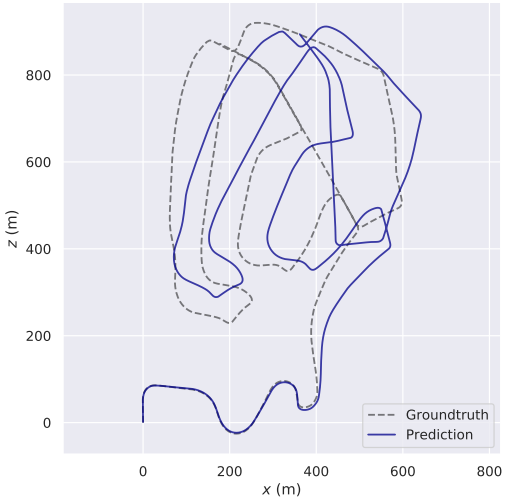
Figure 5.17: The sequence of glimpses' observations for the first frame on sequence 2. The scales are used only to some extent; the glimpse sensor exploits high-gradient regions in most captures.

Testing stage. The baseline RAM-VO was then tested on several sequences, as shown in Figure 5.18; the training sequences are presented in Figure B.1. The results indicate that the RAM-VO learned 6-DoF pose regression; moreover, it could generalize to unseen sequences during training. The predictions generated for the train set show a drift error caused mainly by the rotational component, whereas the translational component impacts most of the predictions on the test set. Their origin is aligned to the ground-truth to highlight the accumulated drift error; the Umeyama method [124] was used only on computing the metrics.

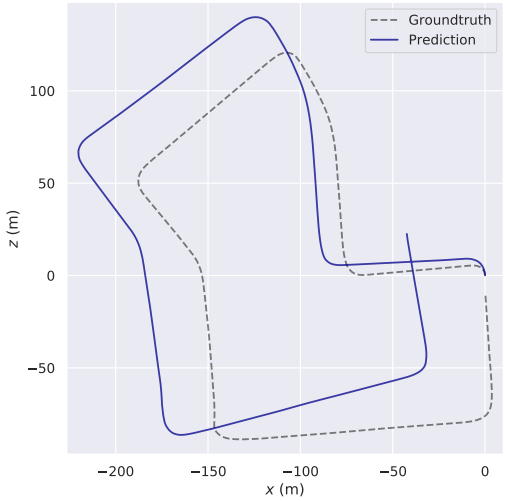


(a) Training sequence 0

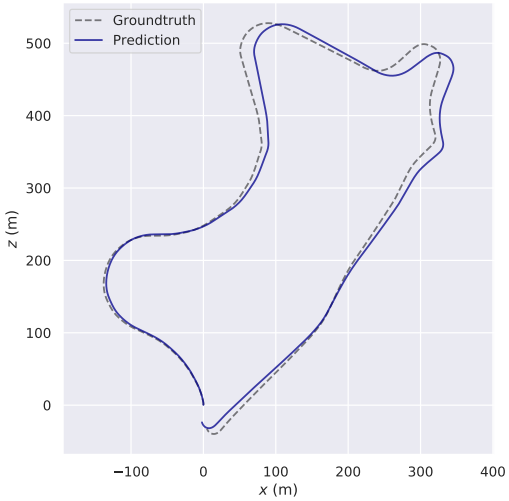
(b) Testing sequence 3



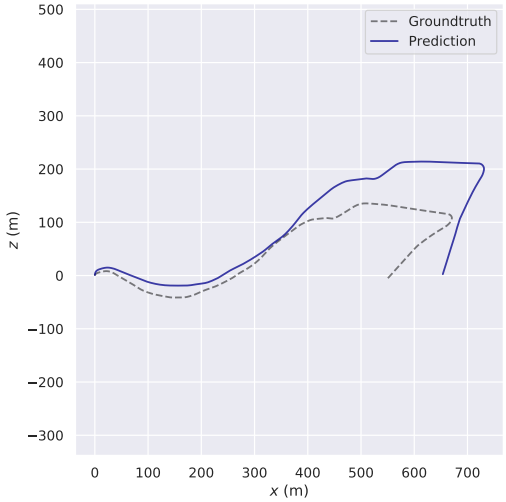
(c) Training sequence 2



(d) Testing sequence 7



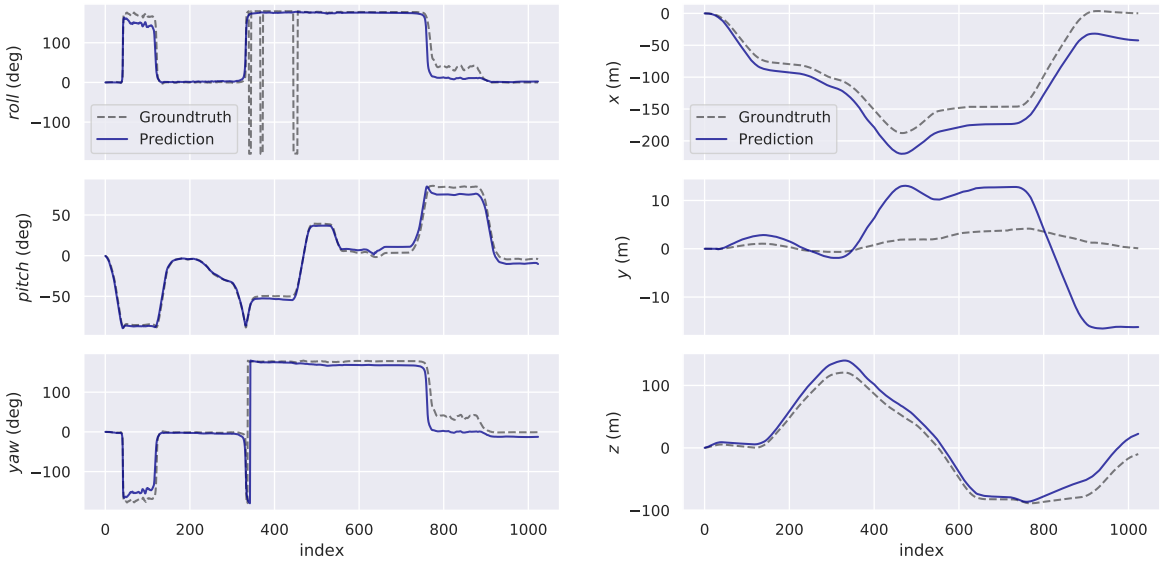
(e) Training sequence 9



(f) Testing sequence 10

Figure 5.18: Trajectory predictions for training and testing sequences on the KITTI dataset. The results indicate the baseline RAM-VO can generalize for unseen sequences.

The prediction's decomposition into the rotational and translational components allows us to investigate in which situations the RAM-VO cannot make a good prediction. For this, Figure 5.19 presents the 6-DoF regression for the entire sequence 7 separated by each component. We can note that only some components have a more significant impact on the drift error, such as the translational component x and z; although component y seems to have a significant error visually, the variation interval is narrow. The translational error occurs mainly when the vehicle also performs a rotation; the RL agent presents difficulty selecting image patches for both the rotational and translation prediction in the same frame pair. Also, abrupt angle variation cannot be predicted adequately.



(a) Rotational components

(b) Translational components

Figure 5.19: The rotational and translational components for sequence 7 on the test set. The translational drift error is significantly higher on the test set.

We conducted several experiments to provide a better understanding of the RAM-VO behavior on complex sequences. Table 5.7 presents a summary for each experiment, including the metrics RPE and ATE for predictions on train and test set. It is important to note that the ATE metric can be misleading, as it evaluates the entire trajectory, so rotation and translation errors might have a compensatory effect on specific trajectories, producing a lower ATE. Therefore, it is preferable to compute relative metrics (RPE) for 100 to 800-meter subsequences and calculate the average; furthermore, computing a separate RPE for rotational and translational components provides a better understanding of the system behavior. In the following experiments, we were mainly interested in validating the number of glimpse's and the RL policy on building a usable latent space for regression. Thus, we vary the glimpse amount from 1 to 12; we also tested configurations with random glimpses. For a detailed comparison by sequence, see Table B.1.

The first experiment consisted of capturing a single image patch at the image's center. The error metrics indicate that the model can overfit the train sequences well, possibly by learning the appearance instead of geometry; however, the ability to generalize is strongly

Configuration	Train set			Test set		
	$\bar{t}_{rpe}(\%)$	$\bar{r}_{rpe}(°)$	$\overline{ATE}$(m)	$\bar{t}_{rpe}(\%)$	$\bar{r}_{rpe}(°)$	$\overline{ATE}$(m)
1 glimpse at center	0.9841	0.4077	3.2162	16.3689	7.6665	36.1037
1 glimpse random	16.5162	4.9401	127.1582	25.9694	7.9394	42.2208
4 glimpses	3.5918	1.6027	36.4627	10.9291	**3.9851**	**17.4179**
8 glimpses	**3.0208**	**1.3928**	**20.3115**	**10.8881**	4.2062	19.8061
8 glimpses random	5.2271	2.1264	58.1929	12.4608	4.1275	21.0045
12 glimpses	3.3349	1.5039	32.5167	13.1815	5.7713	24.7622

Table 5.7: The impact of glimpses on the average RPE and ATE for all sequences on train and test set. $\bar{t}_{rpe}$ represents the average translational RMSE drift (%) on length of 100m to 800m. $\bar{r}_{rpe}$ is the average rotational RMSE drift (°/100m) on length of 100m to 800m. $\overline{ATE}$ represents the average absolute trajectory error.

affected. The information captured from only a central region facilitates the learning of the scale since it tends to be constant for most frames; however, some frames show different behaviors (e.g., obstructions by other vehicles, reflections) and quickly degrades the accumulated trajectory prediction. Also, only one capture in the same location reduces the data diversity necessary to learn complex regressions; the model may memorize the single patch instead of learning a general dynamic. The second experiment consisted of a single capture in a random location; we can see that the training and the generalization are harmed. Although a single random glimpse provides more diversity during training, it tends to capture little and sparse information, which affects the scale learning and makes robust predictions difficult. Random glimpses require general dynamics learning since the input space is ample and the model's capacity is limited. It is important to highlight that these experiments do not use reinforcement learning since the location is already determined. One evident conclusion from these experiments is that a single glimpse does not provide enough information to generalize for unseen sequences.

The subsequent experiments aim to determine the impact of the information's amount on the regression quality; from this, we extend our knowledge to understand the policy's capacity to select informative image regions and the core network's ability to integrate them. Therefore, the experiments consisted of setting the glimpses amount to 4, 8, and 12; all of them determined by the learned policy. The best generalization results were achieved with 4 and 8 glimpses; 12 glimpses have not provided better results, as more data not necessarily means better predictions since the extra observations demand more from the core network and the RL agent – a single poor observation can harm the entire latent space. In the train set, 8 glimpses provided the smallest error for all metrics, followed by the 12 glimpses; however, 4 glimpses present slightly better generalization results than 8 glimpses in the test set. There is a significant gap between train and test results, showed mainly by the relative metrics.

The experiment with 8 glimpses provides the lowest error during training; however, the results are still close to those presented by an 8-glimpse random policy for testing. The RL agent's difficulty in learning a better-than-average policy probably happens due to a) the REINFORCE [130] algorithm presents difficulties in building a decent latent space with information from rotation and translation; b) the state representation makes

the differentiation between states difficult, directly impacting the best action selection. Therefore, the following sections will investigate the impact of replacing the REINFORCE algorithm with Proximal Policy Optimization (PPO) [111], and improving the state representation by adding optical flow as contextual information to initialize the RL agent and also adding temporal information from the last frames.

5.2.1 Learning a Policy with Proximal Policy Optimization (PPO)

The REINFORCE [130] algorithm is known for presenting converge issues and slowness; this occurs by sudden updates on the policy's parameters, which can harm the entire training by converging to suboptimal solutions and losing the entropy quickly – we desire an algorithm that explores the state/action possibilities in the beginning, slowly converging to the best action selection as the training reaches its end. Another significant issue occurs when using small mini-batches during training, which increases the variance across episodes, making the convergence very slow. A baseliner network attenuates this variance by learning the state value, but we can do it better by using a fully actor-critic algorithm.

The Proximal Policy Optimization (PPO) [111] aims to solve these problems by updating the policy inside trusted regions. Therefore, the PPO's surrogate function determines that the current policy must be close to the last one, avoiding large parameters shift. Another essential feature employed in our method is the maximization of the entropy during training – this feature is primarily used in maximum entropy methods and aims to prevent the algorithm from quickly converging to local minima. The use of memory replay also played an essential role since it enabled the policy refinement with already sampled data, improving the architecture's efficiency. We defined the refinement iteration as 20, which means the RL policy updates in the proportion of 20:1, compared to the supervised network. This is an important advancement since we always want the best policy to control the input information flow.

In practice, the PPO implementation consisted of replacing the locator and baseliner network with a similar architecture in terms of layers and hidden units, as detailed in Table A.4. RAM-VO with PPO learned a different policy, although very similar to the one learned by the REINFORCE algorithm. From the heatmap of all glimpses' fixation, as shown in Figure 5.20, we can observe the Gaussian pattern is still present, but with a lower standard deviation.

We investigated the gap between train and test results through several experiments. We hypothesize that the core network may have a larger capacity than the necessary for a good state representation; this fact can harm the generalization since lowering the capacity tends to promote the learning of better representations. The core network corresponds to most of the model's parameters; therefore, knowing the minimum capacity required to achieve good results is crucial for delivering lightweight models. We captured eight glimpses for all experiments to determine the impact of replacing REINFORCE by PPO. Table 5.8 show the experiments performed by varying the core network's hidden units from 1024 to 512 and 256. For a detailed comparison by sequence, see Table B.2. The chosen configurations generated architectures with large variance in parameters; therefore, we can observe the capacity's impact on the result's quality and computational cost.

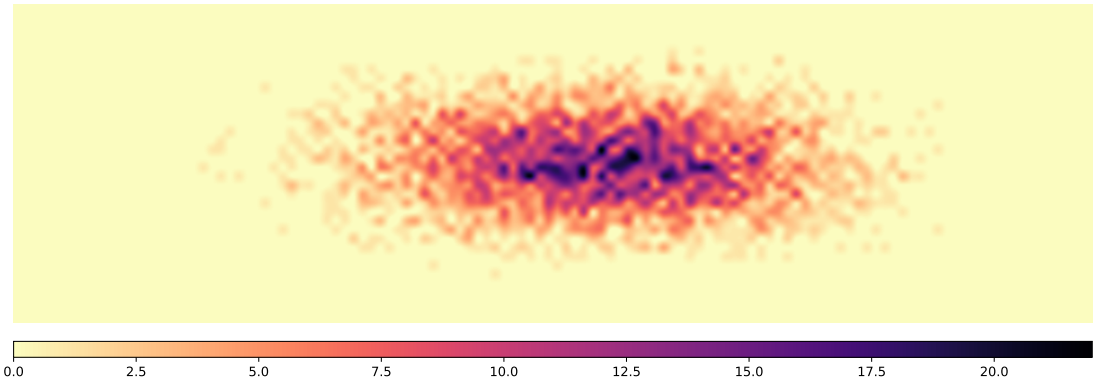


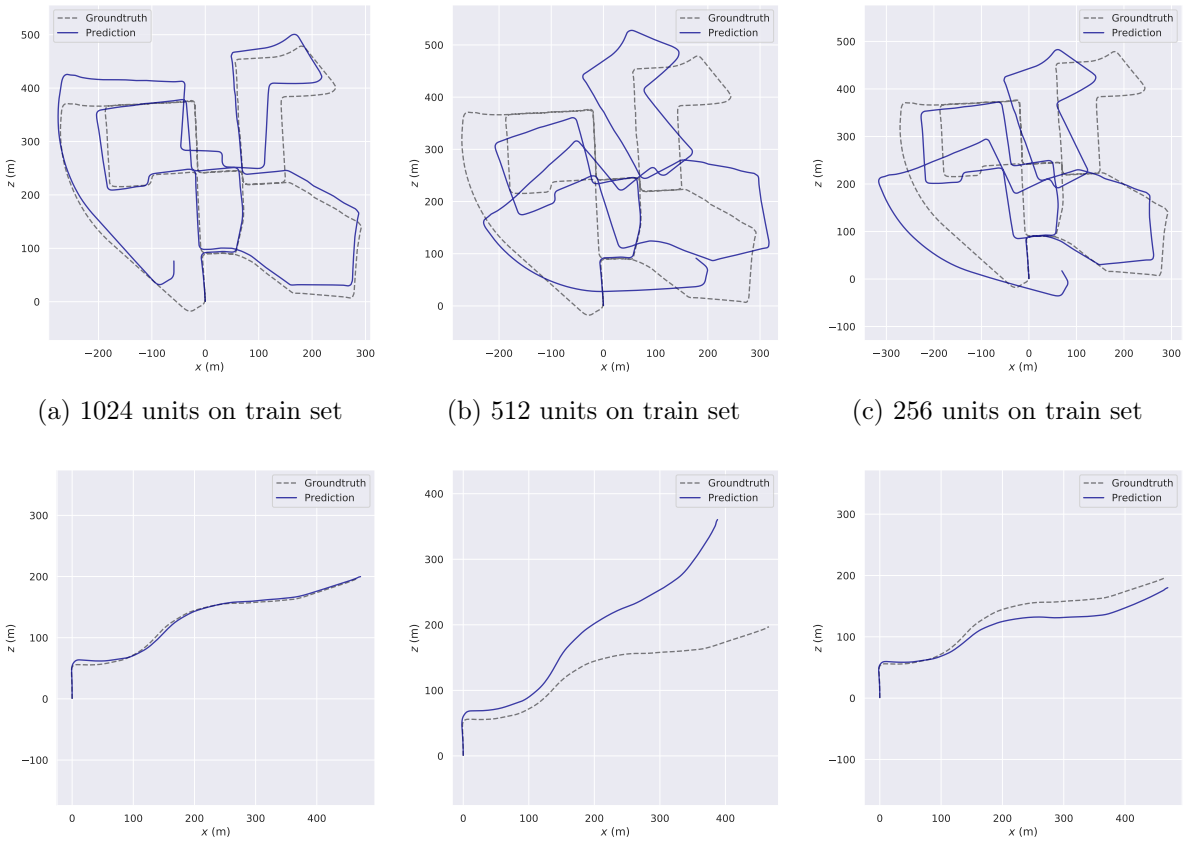
Figure 5.20: The heatmap for all glimpses' fixation on the test sequence 3. The learned PPO policy presents similar behavior as those presented by the REINFORCE. The heatmap is generated with the bicubic interpolation.

		Train set			**Test set**		
Config.	**Param.**	$\bar{t}_{rpe}(\%)$	$\bar{r}_{rpe}(^\circ)$	$\overline{ATE}$(m)	$\bar{t}_{rpe}(\%)$	$\bar{r}_{rpe}(^\circ)$	$\overline{ATE}$(m)
1024 REINF	17.35M	**3.0208**	**1.3928**	**20.3115**	10.8881	4.2062	19.8061
1024 PPO	16.54M	3.8720	1.5470	28.0067	**9.5815**	**3.9858**	**14.7036**
512 PPO	5.84M	5.2254	2.1378	40.8940	13.8468	6.6393	21.6508
256 PPO	2.92M	5.3610	2.2898	27.9853	9.8839	4.4774	16.1354

Table 5.8: The impact of model's capacity on the average RPE and ATE for all sequences on train and test set. $\bar{t}_{rpe}$ represents the average translational RMSE drift (%) on length of 100m to 800m. $\bar{r}_{rpe}$ is the average rotational RMSE drift ($^\circ$/100m) on length of 100m to 800m. $\overline{ATE}$ represents the average absolute trajectory error.

We conclude, from the experiments, that RAM-VO achieved the best generalization results with 1024 hidden units and PPO. Since the only change was the REINFORCE replacement by the PPO algorithm. We attribute the better generalization to learning a more robust policy. This improvement, although subtle, allows us to achieve visually better results, as shown by Figure 5.21. The other experiments aim to determine the impact of lower capacity models on the regressions' quality. We can observe that decreasing the number of parameters increases the relative error during training; however, the generalization is not strongly affected by the model's capacity when evaluated in the test set. Also, the 512-hidden-unit model performed worse than the 256 on the test; therefore, we conclude that the states' representation capacity does not significantly influence the generalization.

The 1024-hidden-unit models have the best training and testing performance; however, the difference in the results' quality may not justify a three-fold increase in the parameters amount. The RAM-VO version with 2.92 million parameters provided results compatible with the 16.54-million version in the test; in contrast, similar methods easily pass 32 million parameters [126]. Deep RL tends to overfit the policy in training, having significant issues with generalizing – we noted this behavior in our experiments. Increasing the input data and model's capacity tends to improve results in training but does not affect generalization significantly. In conclusion, we can assert that PPO provided a slightly better generalization capacity than the REINFORCE algorithm – partly proving our



(a) 1024 units on train set (b) 512 units on train set (c) 256 units on train set

(d) 1024 units on test set (e) 512 units on test set (f) 256 units on test set

Figure 5.21: The impact of varying the core network's hidden units on trajectory predictions for train and test sets.

hypotheses H_3 in which PPO can generate a useful latent space for regression, but, from the experiments, we cannot affirm that the learned policy is more robust.

5.2.2 Providing Optical Flow as Contextual Information

Following our hypothesis that the RL states are poorly represented, we decided to initialize the last LSTM layer in the core network with contextual information, as suggested in the DRAM work [2]. The most helpful information is the optical flow extracted between the two frames since it resumes the salient features required to predict the motion. We decided to inform the dense optical flow extracted by the Farneback [33] method and use CNN layers to determine their importance, as shown in Figure 5.22. Initializing the RL agent with the dense optical enables it to determine the most valuable image regions for further exploration through the subsequent glimpses; therefore, the optical flow is used to provide an overview of the scene so that the agent can optimize the subsequent observations.

However, the initialization must occur only for the last LSTM hidden space $\mathbf{h}_t^2$; the regressor network must be changed to consume the first LSTM latent space $\mathbf{h}_t^1$. The locator and baseliner networks still consume the $\mathbf{h}_t^2$. These alterations aim to prevent the regressor from shortcutting the observations' integration and learning directly from the contextual information. The context network is then trained with gradients provided

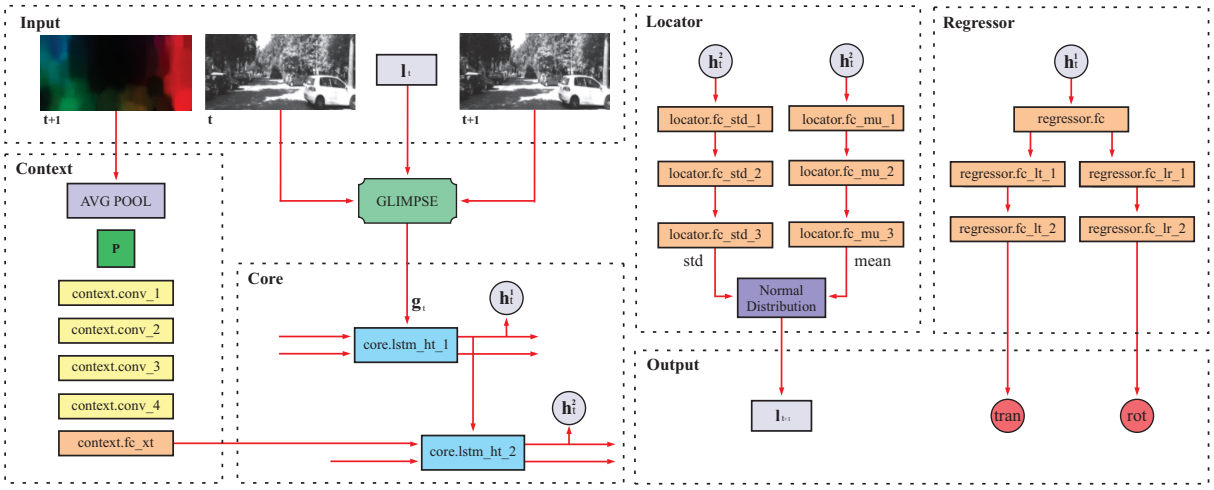


Figure 5.22: The RAM-VO architecture with contextual information. The modifications included adding a secondary network to provide contextual information from the dense optical flow. The context is used to initialize the RL state and give a scene overview for further observations. The baseliner network was omitted.

by the baseliner network; thus, the goal is to maximize the expected reward. The entire process consists of extracting the optical flow from the input image pair, resizing it to a determined size by an average pool operation, and processing it through four convolutional layers with four channels. Similar to the glimpse network, we do not employ pooling operations between layers. For more information on the model's configuration, see Table A.5.

The context network can also be seen as a primitive bottom-up attentional system, where the salient features present on the scene are informed to the agent a priori. Afterwards, the agent captures more details guided by cognitive processes, in this case, represented by the locator network; this process is called top-down attention. In this sense, contextual information provides a second kind of attention to the RAM-VO; and the LSTM latent space initialization provides a way to embed information for the first RL state. This allows the agent to have a scene overview of interesting regions and determine the first glimpse location instead of randomly chosen.

Also, the optical flow allows the RL agent to determine whether the vehicle is either in rotational or translational motion so that the integration of the observations can occur appropriately. The optical flow can be represented as an image, where the flow's magnitude and direction are represented in independent channels, as shown in Figure 5.23. The Farneback [33] method generates an image pyramid with increasing resolutions, which enables tracking large object's motion; however, more pyramid levels increase the computation cost – we keep it as 10 with a window size of 40x40 pixels. The motion is tracked from the lowest-resolution level, and it is refined as the keypoints are propagated to the next level. Sparse optical flow can also be used, but a different structure must be implemented to deal with the features' representation.

The RAM-VO results achieved with contextual information are shown in Table 5.9. For a comparison by sequences see Table B.3. Both relational and absolute pose errors have diminished compared to the RAM-VO without context; however, the generalization

(a) Translation motion on frame 0.

(b) Translation motion on frame 36.

(c) Rotation motion on frame 112.

(d) Rotation motion on frame 122.

Figure 5.23: Dense optical flow extracted by the Farneback [33] method for rotational and translational motions on sequence 0. We can observe that the motion dynamics are different: rotational information tends to be present on the entire image, while the translational are mainly concentrated around the borders.

results have not shown any improvement. The better results achieved in training may not justify the addition in complexity and computation cost; the context network added almost 5 million parameters to the baseline model.

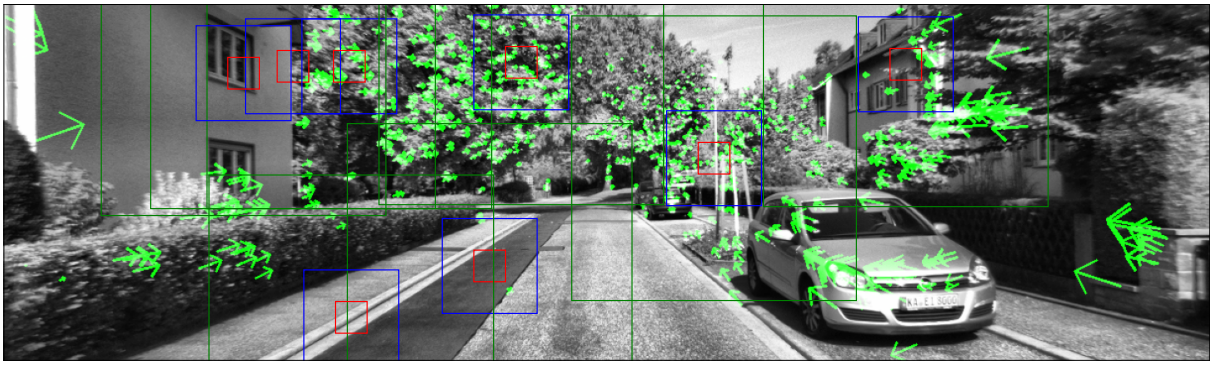
	Train set			**Test set**		
Config.	$\bar{t}_{rpe}(\%)$	$\bar{r}_{rpe}(°)$	$\overline{ATE}$(m)	$\bar{t}_{rpe}(\%)$	$\bar{r}_{rpe}(°)$	$\overline{ATE}$(m)
Baseline	3.0208	1.3928	20.3115	**10.8881**	**4.2062**	**19.8061**
Baseline w/ context	**2.5410**	**1.1668**	**17.3839**	11.7558	5.3126	23.7073

Table 5.9: The impact of contextual information on the average RPE and ATE for all sequences on train and test set. $\bar{t}_{rpe}$ represents the average translational RMSE drift (%) on length of 100m to 800m. $\bar{r}_{rpe}$ is the average rotational RMSE drift (°/100m) on length of 100m to 800m. $\overline{ATE}$ represents the average absolute trajectory error.

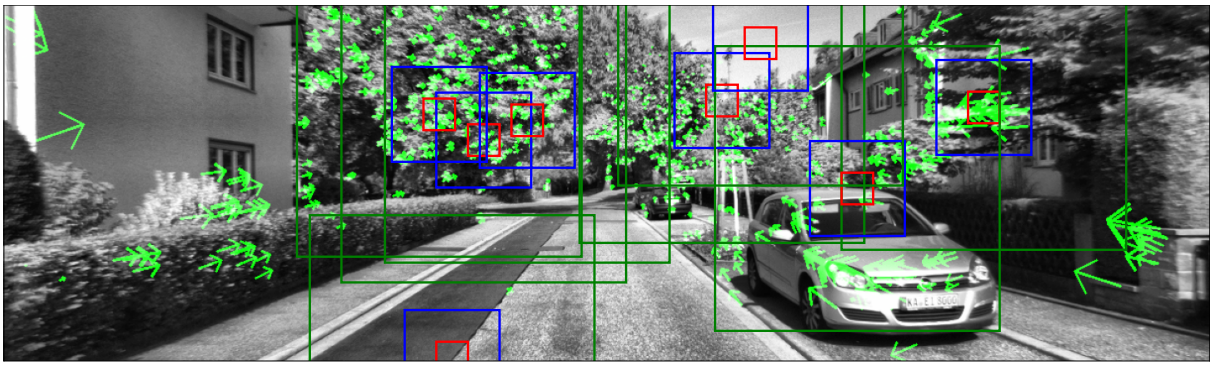
The RL agent's initialization with optical flow tends to generate policies that explore regions where motion is more apparent, as seen in Figure 5.24. We can observe that the totality of 8 glimpses cover central regions where mainly rotation is present; also, the scales are more explored than the baseline version. Important to highlight that several hyper-parameters determine the sparse optical flow extraction; therefore, even regions where significant motion is not identified may present important information for regression. In conclusion, optical flow can be used to initialize the RL agent and achieve better results, at least in training – proving our hypothesis H_4.

5.2.3 Learning Sequential Information

Although RAM-VO can generalize for unseen sequences, some trajectories accumulate a significant drift error. One way to attenuate this is to incorporate temporal information instead of only training with random image pairs. Informing previous frames than the current one allows the network to determine what kind of motion is in progress; this



(a) The baseline RAM-VO.



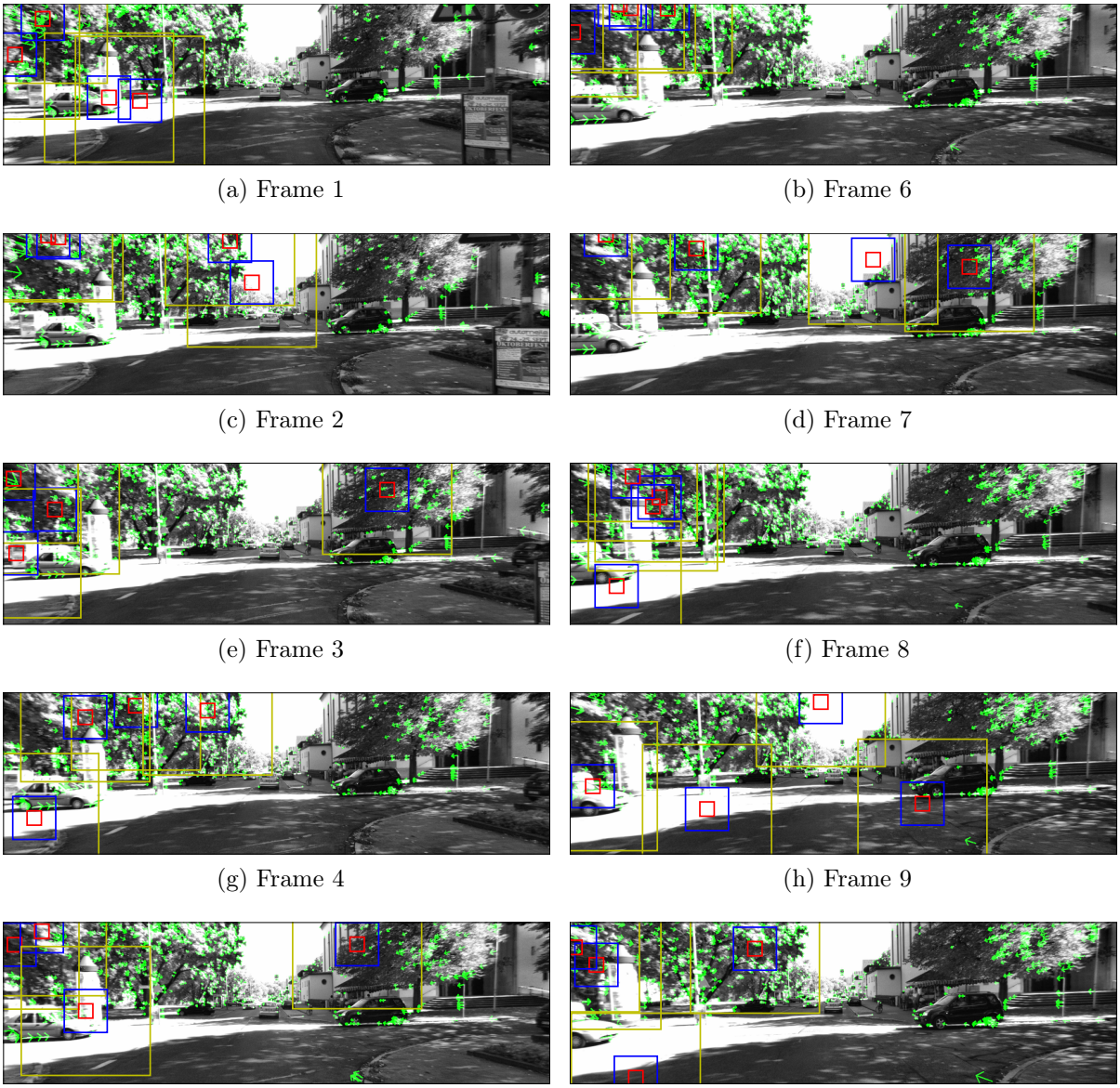
(b) RAM-VO with contextual information.

Figure 5.24: The sparse optical flow with eight glimpses captures for the first frame pair on test sequence 3.

generates a more robust representation of the RL state, making it possible to improve differentiation between states and, consequently, choose the best action.

In this sense, we propose two experiments with substantial differences between them. The first experiment consists of adding two previous frames, totaling four frames, and inserting them in the channels of convolutional operations. The second experiment consists of distributing the glimpses between the four frames so that the first pair of frames receive four glimpses and the current one receives the other four glimpses; in this case, the agent's environment has been extended. The glimpses' observations for the first ten frames in sequence 8 with the 2-2-sequential model is shown in Figure 5.25, indicating that the model tends to cover on average regions of high magnitude in optical flow.

The experiment with four frames on the CNN channels has not shown any improvements; instead, it increased the translational error significantly. The geometry learning seems to be more accurate when applied only to two frames, and four frames seem to have caused more losses in the motion representation. The experiment with 2-2 sequential frames, although it has not achieved better results, it showed competitive results compared to the baseline. We considered that improving the RL state by increasing the environment's information is a better way to incorporate sequential information than on the CNN channels. The results of the experiments are shown in Table 5.10. For a complete comparison by sequence, see Table B.4.



(a) Frame 1 (b) Frame 6

(c) Frame 2 (d) Frame 7

(e) Frame 3 (f) Frame 8

(g) Frame 4 (h) Frame 9

(i) Frame 5 (j) Frame 10

Figure 5.25: The sequence of glimpses' observations for the first 10 frames on sequence 8 with optical flow. The glimpses seem to have a predilection for cover regions with high magnitude in optical flow.

	Train set			Test set		
Config.	$\bar{t}_{rpe}(\%)$	$\bar{r}_{rpe}(^\circ)$	$\overline{ATE}$(m)	$\bar{t}_{rpe}(\%)$	$\bar{r}_{rpe}(^\circ)$	$\overline{ATE}$(m)
Baseline	**3.0208**	**1.3928**	20.3115	**10.8881**	**4.2062**	**19.8061**
4 frames on channels	6.0316	2.4514	56.8278	15.0731	7.4456	26.8214
2-2 frames sequential	3.5017	1.6121	**15.6343**	12.7082	5.5148	22.4577

Table 5.10: The impact of sequential information on the average RPE and ATE for all sequences on train and test set. $\bar{t}_{rpe}$ represents the average translational RMSE drift (%) on length of 100m to 800m. $\bar{r}_{rpe}$ is the average rotational RMSE drift (°/100m) on length of 100m to 800m. $\overline{ATE}$ represents the average absolute trajectory error.

5.3 Comparison with Literature

This section presents a brief literature comparison between RAM-VO and similar methods. For this purpose, we selected the model with the best generalization results. In this case, the RAM-VO with PPO will be compared to ORB-SLAM [87], DeepVO [126], and ESP-VO [127], as shown in Table 5.11. All selected methods perform monocular visual odometry in the KITTI dataset. Except for the ORB-SLAM method, the others are end-to-end learning methods.

Method	Input data(%)	Seq. 03		Seq. 07		Seq. 10	
		$t_{rpe}(\%)$	$r_{rpe}(°)$	$t_{rpe}(\%)$	$r_{rpe}(°)$	$t_{rpe}(\%)$	$r_{rpe}(°)$
ORB-SLAM [87]	-	21.07	18.36	24.53	38.90	86.51	98.90
DeepVO [126]	100.00	8.49	6.89	3.91	4.60	**8.11**	8.83
ESP-VO [127]	100.00	6.72	6.46	**3.52**	5.02	9.77	10.20
RAM-VO 1024	**5.68**	**5.72**	**3.08**	9.17	5.63	13.85	**3.24**
RAM-VO 256	**5.68**	7.08	4.01	7.55	**4.30**	15.02	5.12

Table 5.11: RAM-VO results compared with other methods on test sequences. t_{rpe} is the average translational RMSE drift (%) on length of 100m to 800m. r_{rpe} is the average rotational RMSE drift (°/100m) on length of 100m to 800m. RAM-VO presents comparable results with significantly fewer parameters and input information consumption.

RAM-VO obtained competitive results using less input information than similar methods, around 5.7% of the total available (considering the eight glimpses with the size of 32x32 pixels). While RAM-VO uses top-down attention to capture regions of interest, methods like ORB-SLAM need to analyze an entire image to detect keypoints. Besides, ORB-SLAM is a geometric method that depends on high-texture regions for an accurate keypoint match between frames. In the same sense, the results provided by ORB-SLAM have some improvements over pure odometry, such as bundle adjustment – which makes a fair comparison problematic; however, it enables us to analyze the difference between learning-based and geometry-based methods.

DeepVO and ESP-VO are both based on the FlowNet [36] architecture and therefore have more convolutional layers and channels than RAM-VO. These architectures perform direct visual odometry by determining the frames' correspondence from the pixel values; therefore, they are more robust to outliers than ORB-SLAM. However, direct methods are costly, especially DeepVO and ESP-VO, which use the entire image as input. In visual odometry, the motion is present in the entire image; hence, more data does not necessarily bring novel information. Nevertheless, an efficient image patch selection and integration become necessary since inappropriate regions could be selected (e.g., other vehicles, walking people, low-contrast and high-reflective regions) – with RAM-VO, we demonstrate this assumption to some extent.

As observed during the architecture's construction, RAM-VO presents difficulty obtaining the world scale and determining the translational motion with greater precision. This issue probably happens due to the small image patches that, depending on the vehicle's velocity, can prevent the overlap between frames, compromising the regression since

it relies on the features' correspondence. The other methods can achieve better results concerning the translational component because they do not impose any restriction on the image size, allowing more overlap across frames.

The model's capacity is also a relevant fact to be considered. Although DeepVO and ESP-VO may present similar results on average, the RAM-VO with 256 hidden units in the core network achieves comparable results with only 2.7 million parameters in total – which is much lower than the 17 million parameters reached by the RAM-VO with 1024 hidden units. Concurrent methods regularly pass 32 million parameters, especially when they are extensions of architectures like AlexNet [65], and FlowNet [36]. Also, these CNN networks are considerably deep and add a high cost in terms of floating-point operations, making the training slow and the deployment a complicated task.

In the same way, the results presented by the ORB-SLAM are worse compared to learning-based methods, but they are sufficient to build online applications on mobile devices. Learning-based methods tend to be slow and costly to run due to requiring high-end devices with large processing power, GPUs, and better batteries; all these issues make adopting learning methods problematic. In this context, RAM-VO represents an alternative method capable of providing results similar to large models but with a smaller cost in terms of trainable parameters and floating-point operations.

Chapter 6

Conclusion and Future Work

In this work, we proposed the RAM-VO architecture for end-to-end visual odometry regression using only monocular images. RAM-VO is extended from RAM [80] architecture and therefore implements visual attentional concepts for an optimized selection of input information; the image's observations are guided by reinforcement learning and integrated on a recurrent basis. To the best of our knowledge, RAM-VO is the first architecture for visual odometry that implements reinforcement learning in part of the pipeline. The experimental results indicate that RAM-VO can predict 6-DoF poses in the real-world KITTI [42] dataset with moderate generalization for unseen sequences.

Initially, the RAM architecture was extended to regression tasks in simple problems such as the Pixel dataset; after that, more sophisticated spatial (CNN) and temporal (LSTM) elements were included to improve the prediction in complex visual scenarios as the City dataset. Developing our datasets allowed us to incrementally increase the problems' complexity, add the pertinent structures to address them efficiently, and adapt the RAM to visual regression. From these modifications, we proved our hypotheses H_1 and H_2, in which RAM was able to perform complex and straightforward regressions using visual information selected by an attentional mechanism.

Then, we proposed the RAM-VO architecture, which performed visual odometry in the KITTI dataset. RAM-VO is fundamentally more complex than the previous architectures since 6-DoF pose regressions are tricky, and the input images have increased in resolution. Thus, we increased the model's representation capacity and treated the input images in the CNN channels to favor learning the scene geometry instead of appearance. The best training and generalization results were achieved with 4-8 glimpses; more glimpses and random policies have not provided efficient learning. Therefore, we conclude that the REINFORCE [130] algorithm can learn a policy able to produce a usable latent space for visual odometry, especially in the training set.

To investigate the performance gap between training and test results, we replaced the REINFORCE algorithm with the Proximal Policy Optimization (PPO) [111], which learned a visually similar policy, but improved the learning curve, prevented the model from collapsing in some situations, and increased the model's generalization capacity substantially. Therefore, the hypothesis H_3 was partially proved, PPO algorithm produced a usable latent space for visual odometry, but we cannot affirm that the learned policy is more robust than the one learned by REINFORCE. Also, we investigated the model's

cost in terms of trainable parameters; we conclude that it is possible to achieve comparable results between a 17-million-parameter and a 2-million-parameter model. Therefore, capacity does not play a significant role in generalization; instead, the input/state representation and the reward function should be investigated further.

We also carried out experiments to investigate the impact of producing RL states more representative for the regression. Therefore, we added the dense optical flow as contextual information to initialize the RL agent; contextual information helped the RL agent to infer the motion present in the scene better, and consequently, determine the subsequent observations with greater precision. The hypothesis H_4 was then partially proved since the optical flow improved the results in training but had a low impact in generalization. In the same sense, RAM-VO was adapted to consume sequential frames in the expectation of having a better state representation; however, no significant impact on the predictions was observed.

The comparison with the literature indicated that RAM-VO could achieve competitive results compared to similar methods using significantly less trainable parameters and input information, proving our hypothesis H_5. Similar learning methods use the whole input image to determine the pose, while the RAM-VO can generate regressions with a small fraction, around 5.7% of the whole available input data. RAM-VO was especially able to predict rotational motion, being consistently better than similar methods; for translational motion, it achieves comparable results. Therefore, considering the ever-growing demand for robust and efficient methods, RAM-VO is another step towards this end.

For future work, we can highlight the improvement of the model's generalization through data augmentation and training with alternative datasets, in which the sequences provide different dynamics, mainly for the vehicle speed. Another critical point is to improve the model's temporal representation with the possibility of adding external memories to keep information between distant frames – this would significantly impact the drift error. Regularization techniques such as dropout and batch normalization should be carefully investigated as a way to improve generalization. The reward function should be explored in-depth; we believe that reward shaping can induce better policies.

The proposed architecture does not implement any method for correcting the trajectory, such as local bundle adjustment. Therefore, the trajectories presented are only computed considering the incremental pose's displacement; RAM-VO can be extended to include these features and reduce the drift error. Besides, inertial information such as IMU can improve the translational motion; however, sensor fusion must be carefully thought out. RAM-VO can be adapted to consume stereo and RGB-D images, which significantly impact the depth prediction. Similarly, inverse depth maps can be generated by unsupervised learning and help to improve mainly the translational motion. One valuable contribution would be generating depth maps from the small patches captured by the glimpse network – currently, the depth maps computation is costly and considers the whole image.

Self-attention is an essential step towards reducing the computational cost, primarily if implemented in an end-to-end manner. Self-attention can be employed at three levels in RAM-VO: a) replacing the convolutional networks and integrating the pixels of the different glimpse scales; b) integrating the latent space of the two sensor glimpses; c)

integrating the glimpses observations in the core network. Thus, the use of self-attention would completely replace the need for CNN and LSTM structures. In the same line, other attentional mechanisms can be implemented in other parts of the pipeline, such as between the actor and the critic networks, and generate the predictions. The association of reinforcement learning, attention, and memory elements provides a robust framework to build cognitive architectures.

Learning the motion dynamics presented in visual odometry tends to be challenging when training from scratch; further, the available datasets present few examples of the many nonlinearities – learning complex regressions and optimal policies require as many samples as possible. Therefore, transfer learning can be employed to reduce the training cost and to provide generic dynamics. Curriculum learning is another important feature that could improve the learning of better policies by incrementally increase the regression difficulty – which could be done by starting the regression with only one component and adding the others on the way. Meta-learning can also be an essential tool for learning generic dynamics across many datasets, thus, solving the limitations presented by a single dataset.

Finally, RAM-VO can be employed in many similar tasks, and a natural extension is to use in Simultaneous Localization and Mapping (SLAM) and the other Structure from Motion (SfM) techniques. Other domains can benefit from RAM-VO characteristics, such as part-based models, face recognition systems, and image-to-text labeling. Also, RAM-VO can easily be adapted to perform classification and to consume other kinds of input data. The attentional system is ideal for integrating multi-modal data and extracting meaningful representations; the reinforcement learning paradigm also allows the system to find the most beneficial behavior for the task at hand.

Bibliography

[1] Yasin Almalioglu, Muhamad Risqi U. Saputra, Pedro P. B. de Gusmao, Andrew Markham, and Niki Trigoni. GANVO: Unsupervised Deep Monocular Visual Odometry and Depth Estimation with Generative Adversarial Networks. *arXiv:1809.05786 [cs, stat]*, September 2018. arXiv: 1809.05786.

[2] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014.

[3] Hernán Badino, Akihiro Yamamoto, and Takeo Kanade. Visual odometry by multi-frame feature integration. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 222–229, 2013.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[5] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[6] Houssem Eddine Benseddik, Oualid Djekoune, and Mahmoud Belhocine. Sift and surf performance evaluation for mobile robot-monocular visual odometry. *Journal of Image and Graphics*, 2(1):70–76, 2014.

[7] Jose-Luis Blanco-Claraco, Francisco-Angel Moreno-Duenas, and Javier Gonzalez-Jimenez. The Málaga urban dataset: High-rate stereo and LiDAR in a realistic urban scenario. *The International Journal of Robotics Research*, 33(2):207–214, February 2014.

[8] Samarth Brahmbhatt, Jinwei Gu, Kihwan Kim, James Hays, and Jan Kautz. Geometry-Aware Learning of Maps for Camera Localization. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2616–2625, Salt Lake City, UT, USA, June 2018. IEEE.

[9] Hudson M. S. Bruno and Esther L. Colombini. A comparative evaluation of learned feature descriptors on hybrid monocular visual slam methods. In *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*, pages 1–6, 2020.

[10] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, September 2016.

[11] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose Neira, Ian Reid, and John J. Leonard. Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age. *IEEE Transactions on Robotics*, 32(6):1309–1332, December 2016. arXiv: 1606.05830.

[12] J. Campbell, R. Sukthankar, I. Nourbakhsh, and A. Pahwa. A Robust Visual Odometry and Precipice Detection System Using Consumer-grade Monocular Vision. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 3421–3427, Barcelona, Spain, 2005. IEEE.

[13] Changhao Chen, Stefano Rosa, Yishu Miao, Chris Xiaoxuan Lu, Wei Wu, Andrew Markham, and Niki Trigoni. Selective Sensor Fusion for Neural Visual-Inertial Odometry. *arXiv:1903.01534 [cs]*, March 2019. arXiv: 1903.01534.

[14] Ming-yue Chen, Cai-ling Wang, and Hua-jun Liu. Salient FlowNet and Decoupled LSTM Network for Robust Visual Odometry. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2699–2706, Dali, China, December 2019. IEEE.

[15] Yang Cheng, Mark Maimone, and Larry Matthies. Visual odometry on the mars exploration rovers. In *2005 IEEE International Conference on Systems, Man and Cybernetics*, volume 1, pages 903–910. IEEE, 2005.

[16] Hsiang-Jen Chien, Chen-Chi Chuang, Chia-Yen Chen, and Reinhard Klette. When to use what feature? sift, surf, orb, or a-kaze features for monocular visual odometry. In *2016 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pages 1–6. IEEE, 2016.

[17] Thomas A. Ciarfuglia, Gabriele Costante, Paolo Valigi, and Elisa Ricci. Evaluation of non-geometric methods for visual odometry. *Robotics and Autonomous Systems*, 62(12):1717–1730, December 2014.

[18] Ronald Clark, Sen Wang, Hongkai Wen, Andrew Markham, and Niki Trigoni. VINet: Visual-Inertial Odometry as a Sequence-to-Sequence Learning Problem. *arXiv:1701.08376 [cs]*, January 2017. arXiv: 1701.08376 version: 1.

[19] Lee Clement and Jonathan Kelly. How to Train a CAT: Learning Canonical Appearance Transformations for Direct Visual Localization Under Illumination Change. *IEEE Robotics and Automation Letters*, 3(3):2447–2454, July 2018. arXiv: 1709.03009.

[20] Esther Luna Colombini, A da Silva Simoes, and CHC Ribeiro. *An attentional model for intelligent robotics agents.* PhD thesis, Instituto Tecnológico de Aeronáutica, São José dos Campos, Brazil, 2014.

[21] Andrew I. Comport. *Direct Iterative Closest Point for Real-time Visual Odometry.*

[22] Gabriele Costante, Michele Mancini, Paolo Valigi, and Thomas A. Ciarfuglia. Exploring Representation Learning With CNNs for Frame-to-Frame Ego-Motion Estimation. *IEEE Robotics and Automation Letters*, 1(1):18–25, January 2016.

[23] Igor Cvišić and Ivan Petrović. Stereo odometry based on careful feature selection and tracking. In *2015 European Conference on Mobile Robots (ECMR)*, pages 1–6. IEEE, 2015.

[24] Hamed Damirchi, Rooholla Khorrambakht, and Hamid D Taghirad. Exploring self-attention for visual odometry. *arXiv preprint arXiv:2011.08634*, 2020.

[25] Leonardo De-Maeztu, Unai Elordi, Marcos Nieto, Javier Barandiaran, and Oihana Otaegui. A temporally consistent grid-based visual odometry framework for multi-core architectures. *Journal of Real-Time Image Processing*, 10(4):759–769, 2015.

[26] Alana de Santana Correia and Esther Luna Colombini. Attention, please! a survey of neural attention models in deep learning, 2021.

[27] Jeffrey Delmerico, Titus Cieslewski, Henri Rebecq, Matthias Faessler, and Davide Scaramuzza. Are We Ready for Autonomous Drone Racing? The UZH-FPV Drone Racing Dataset. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6713–6719, Montreal, QC, Canada, May 2019. IEEE.

[28] Christian Dornhege and Alexander Kleiner. Visual Odometry for Tracked Vehicles. page 7.

[29] Felix Endres, Jurgen Hess, Nikolas Engelhard, Jurgen Sturm, Daniel Cremers, and Wolfram Burgard. An evaluation of the RGB-D SLAM system. In *2012 IEEE International Conference on Robotics and Automation*, pages 1691–1696, St Paul, MN, USA, May 2012. IEEE.

[30] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct Sparse Odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625, March 2018.

[31] Jakob Engel, Jurgen Sturm, and Daniel Cremers. Semi-dense Visual Odometry for a Monocular Camera. In *2013 IEEE International Conference on Computer Vision*, pages 1449–1456, Sydney, Australia, December 2013. IEEE.

[32] Jakob Engel, Vladyslav Usenko, and Daniel Cremers. A Photometrically Calibrated Benchmark For Monocular Visual Odometry. *arXiv:1607.02555 [cs]*, October 2016. arXiv: 1607.02555.

[33] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*, pages 363–370. Springer, 2003.

[34] Tuo Feng and Dongbing Gu. SGANVO: Unsupervised Deep Visual Odometry and Depth Estimation with Stacked Generative Adversarial Networks. *IEEE Robotics and Automation Letters*, pages 1–1, 2019. arXiv: 1906.08889.

[35] Mark Fiala and Alex Ufkes. Visual odometry using 3-dimensional video input. In *2011 Canadian Conference on Computer and Robot Vision*, pages 86–93. IEEE, 2011.

[36] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning Optical Flow with Convolutional Networks. *arXiv:1504.06852 [cs]*, May 2015. arXiv: 1504.06852.

[37] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[38] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, Hong Kong, China, May 2014. IEEE.

[39] Wolfgang Förstner. A feature based correspondence algorithm for image matching. *ISPRS ComIII, Rovaniemi*, pages 150–166, 1986.

[40] Friedrich Fraundorfer and Davide Scaramuzza. Visual Odometry : Part II: Matching, Robustness, Optimization, and Applications. *IEEE Robotics & Automation Magazine*, 19(2):78–90, June 2012.

[41] Jianlong Fu, Heliang Zheng, and Tao Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4438–4446, 2017.

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

[43] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[44] Ruben Gomez-Ojeda, Zichao Zhang, Javier Gonzalez-Jimenez, and Davide Scaramuzza. Learning-based Image Enhancement for Visual Odometry in Challenging HDR Environments. *arXiv:1707.01274 [cs]*, April 2018. arXiv: 1707.01274.

[45] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts, 2016.

[46] Michael Grupp. evo: Python package for the evaluation of odometry and slam. `https://github.com/MichaelGrupp/evo`, 2017.

[47] Vitor Guizilini and Fabio Ramos. Semi-parametric learning for visual odometry. *The International Journal of Robotics Research*, 32(5):526–546, April 2013.

[48] Zibin Guo, Mingkun Yang, Ninghao Chen, Zhuoling Xiao, Bo Yan, Shuisheng Lin, and Liang Zhou. LightVO: Lightweight Inertial-Assisted Monocular Visual Odometry with Dense Neural Networks. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Waikoloa, HI, USA, December 2019. IEEE.

[49] Tuomas Haarnoja, Anurag Ajay, Sergey Levine, and Pieter Abbeel. Backprop KF: Learning Discriminative Deterministic State Estimators. *arXiv:1605.07148 [cs]*, September 2017. arXiv: 1605.07148.

[50] Ankur Handa, Thomas Whelan, John McDonald, and Andrew J. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1524–1531, Hong Kong, China, May 2014. IEEE.

[51] Christopher G Harris and JM Pike. 3d positional integration from image sequences. *Image and Vision Computing*, 6(2):87–90, 1988.

[52] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[53] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663, April 2012.

[54] Christopher J. Holder and Toby P. Breckon. Learning to Drive: Using Visual Odometry to Bootstrap Deep Learning for Off-Road Path Prediction. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 2104–2110, Changshu, June 2018. IEEE.

[55] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *JOSA A*, 4(4):629–642, April 1987.

[56] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[57] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259, 1998.

[58] Hailin Jin, Paolo Favaro, and Stefano Soatto. A semi-direct approach to structure from motion. *The Visual Computer*, 19(6):377–394, October 2003.

[59] Daniel Kahneman. *Attention and effort*, volume 1063. Citeseer, 1973.

[60] Christian Kerl, Jurgen Sturm, and Daniel Cremers. Robust odometry estimation for RGB-D cameras. In *2013 IEEE International Conference on Robotics and Automation*, pages 3748–3754, Karlsruhe, Germany, May 2013. IEEE.

[61] Bernd Kitt, Frank Moosmann, and Christoph Stiller. Moving on to dynamic environments: Visual odometry using feature classification. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5551–5556. IEEE, 2010.

[62] Kishore Konda and Roland Memisevic. Unsupervised learning of depth and motion. *arXiv:1312.3429 [cs, stat]*, December 2013. arXiv: 1312.3429.

[63] Kishore Konda and Roland Memisevic. Learning Visual Odometry with a Convolutional Network:. In *Proceedings of the 10th International Conference on Computer Vision Theory and Applications*, pages 486–490, Berlin, Germany, 2015. SCITEPRESS - Science and and Technology Publications.

[64] Alexander S. Koumis, James A. Preiss, and Gaurav S. Sukhatme. Estimating Metric Scale Visual Odometry from Videos using 3D Convolutional Networks. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 265–272, Macau, China, November 2019. IEEE.

[65] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[66] Xin-Yu Kuo, Chien Liu, Kai-Chen Lin, and Chun-Yi Lee. Dynamic attention-based visual odometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 36–37, 2020.

[67] Simon Lacroix, Anthony Mallet, Raja Chatila, and Laurent Gallo. Rover self localization in planetary-like environments. In *Artificial Intelligence, Robotics and Automation in Space*, volume 440, page 433, 1999.

[68] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[69] Stefan Leutenegger, Margarita Chli, and Roland Siegwart. Brisk: Binary robust invariant scalable keypoints. In *2011 IEEE international conference on computer vision (ICCV)*, pages 2548–2555. Ieee, 2011.

[70] Huai-Jen Liang, Nitin J. Sanket, Cornelia Fermüller, and Yiannis Aloimonos. SalientDSO: Bringing Attention to Direct Sparse Odometry. *arXiv:1803.00127 [cs]*, February 2018. arXiv: 1803.00127.

[71] Yimin Lin, Zhaoxiang Liu, Jianfeng Huang, Chaopeng Wang, Guoguang Du, Jinqiang Bai, Shiguo Lian, and Bill Huang. Deep Global-Relative Networks for End-to-End 6-DoF Visual Localization and Odometry. *arXiv:1812.07869 [cs]*, May 2019. arXiv: 1812.07869.

[72] Ning Liu, Yongchao Long, Changqing Zou, Qun Niu, Li Pan, and Hefeng Wu. Adcrowdnet: An attention-injective deformable convolutional network for crowd understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3225–3234, 2019.

[73] Tse-An Liu, Huei-Yung Lin, and Wei-Yang Lin. InertialNet: Toward Robust SLAM via Visual Inertial Measurement. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1311–1316, Auckland, New Zealand, October 2019. IEEE.

[74] H Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133, 1981.

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

[76] András L Majdik, Charles Till, and Davide Scaramuzza. The Zurich urban micro aerial vehicle dataset. *The International Journal of Robotics Research*, 36(3):269–273, March 2017.

[77] Larry Matthies and STEVENA Shafer. Error modeling in stereo navigation. *IEEE Journal on Robotics and Automation*, 3(3):239–248, 1987.

[78] C. Mei, S. Benhimane, E. Malis, and P. Rives. Efficient Homography-Based Tracking and 3-D Reconstruction for Single-Viewpoint Sensors. *IEEE Transactions on Robotics*, 24(6):1352–1364, December 2008.

[79] Maxime Meilland and Andrew I. Comport. On unifying key-frame and voxel-based dense visual SLAM at large scales. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3677–3683, Tokyo, November 2013. IEEE.

[80] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention. *arXiv preprint arXiv:1406.6247*, 2014.

[81] Vikram Mohanty, Shubh Agrawal, Shaswat Datta, Arna Ghosh, Vishnu Dutt Sharma, and Debashish Chakravarty. DeepVO: A Deep Learning approach for Monocular Visual Odometry. *arXiv:1611.06069 [cs]*, November 2016. arXiv: 1611.06069.

[82] N. Molton, A. Davison, and I. Reid. Locally Planar Patch Features for Real-Time Structure from Motion. In *Procedings of the British Machine Vision Conference 2004*, pages 90.1–90.10, Kingston, 2004. British Machine Vision Association.

[83] Hans P Moravec. Techniques towards automatic visual obstacle avoidance. 1977.

[84] Hans P Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical report, Stanford Univ CA Dept of Computer Science, 1980.

[85] Elias Mueggler, Henri Rebecq, Guillermo Gallego, Tobi Delbruck, and Davide Scaramuzza. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM. *The International Journal of Robotics Research*, 36(2):142–149, February 2017.

[86] Peter Muller and Andreas Savakis. Flowdometry: An Optical Flow and Deep Learning Based Approach to Visual Odometry. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 624–631, Santa Rosa, CA, USA, March 2017. IEEE.

[87] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[88] A Nemra, S Slimani, M Bouhamidi, A Bouchloukh, and A Bazoula. Adaptive iterative closest surf for visual scan matching, application to visual odometry. In *3rd International Conference on Systems and Control*, pages 927–932. IEEE, 2013.

[89] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *2011 International Conference on Computer Vision*, pages 2320–2327, November 2011.

[90] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):0756–777, 2004.

[91] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. Ieee, 2004.

[92] Donald A Norman. Toward a theory of memory and attention. *Psychological review*, 75(6):522, 1968.

[93] Clark F Olson, Larry H Matthies, H Schoppers, and Mark W Maimone. Robust stereo ego-motion for long distance navigation. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, volume 2, pages 453–458. IEEE, 2000.

[94] Valentin Peretroukhin, Lee Clement, and Jonathan Kelly. Inferring sun direction to improve visual odometry: A deep learning approach:. *The International Journal of Robotics Research*, January 2018.

[95] Valentin Peretroukhin and Jonathan Kelly. DPC-Net: Deep Pose Correction for Visual Localization. *IEEE Robotics and Automation Letters*, 3(3):2424–2431, July 2018.

[96] Bernd Pfrommer, Nitin Sanket, Kostas Daniilidis, and Jonas Cleveland. PennCOSYVIO: A challenging Visual Inertial Odometry benchmark. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3847–3854, Singapore, Singapore, May 2017. IEEE.

[97] R Hans Phaf, AHC Van der Heijden, and Patrick TW Hudson. Slam: A connectionist model for attention in visual selection tasks. *Cognitive psychology*, 22(3):273–341, 1990.

[98] Alberto Pretto, Emanuele Menegatti, Maren Bennewitz, Wolfram Burgard, and Enrico Pagello. A visual odometry framework robust to motion blur. In *2009 IEEE International Conference on Robotics and Automation*, pages 2250–2257. IEEE, 2009.

[99] Alberto Pretto, Emanuele Menegatti, and Enrico Pagello. Omnidirectional dense large-scale mapping and navigation based on meaningful triangulation. In *2011 IEEE International Conference on Robotics and Automation*, pages 3289–3296. IEEE, 2011.

[100] Richard Roberts, Hai Nguyen, Niyant Krishnamurthi, and Tucker Balch. Memory-based learning for visual odometry. In *2008 IEEE International Conference on Robotics and Automation*, pages 47–52, May 2008. ISSN: 1050-4729.

[101] Xiaogang Ruan, Fei Wang, and Jing Huang. Relative Pose Estimation of Visual SLAM Based on Convolutional Neural Networks. In *2019 Chinese Control Conference (CCC)*, pages 8827–8832, Guangzhou, China, July 2019. IEEE.

[102] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R Bradski. Orb: An efficient alternative to sift or surf. In *ICCV*, volume 11, page 2. Citeseer, 2011.

[103] Muhamad Risqi U. Saputra, Pedro P. B. de Gusmao, Yasin Almalioglu, Andrew Markham, and Niki Trigoni. Distilling Knowledge From a Deep Pose Regressor Network. *arXiv:1908.00858 [cs]*, August 2019. arXiv: 1908.00858.

[104] Muhamad Risqi U. Saputra, Pedro P. B. de Gusmao, Chris Xiaoxuan Lu, Yasin Almalioglu, Stefano Rosa, Changhao Chen, Johan Wahlström, Wei Wang, Andrew Markham, and Niki Trigoni. DeepTIO: A Deep Thermal-Inertial Odometry with Visual Hallucination. *arXiv:1909.07231 [cs]*, January 2020. arXiv: 1909.07231.

[105] Muhamad Risqi U. Saputra, Pedro P. B. de Gusmao, Sen Wang, Andrew Markham, and Niki Trigoni. Learning Monocular Visual Odometry through Geometry-Aware Curriculum Learning. *arXiv:1903.10543 [cs]*, November 2019. arXiv: 1903.10543.

[106] D. Scaramuzza and R. Siegwart. Appearance-Guided Monocular Omnidirectional Visual Odometry for Outdoor Ground Vehicles. *IEEE Transactions on Robotics*, 24(5):1015–1026, October 2008.

[107] Davide Scaramuzza and Friedrich Fraundorfer. Visual Odometry [Tutorial]. *IEEE Robotics & Automation Magazine*, 18(4):80–92, December 2011.

[108] Davide Scaramuzza and Roland Siegwart. Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles. *IEEE transactions on robotics*, 24(5):1015–1026, 2008.

[109] David Schubert, Thore Goll, Nikolaus Demmel, Vladyslav Usenko, Jörg Stückler, and Daniel Cremers. The TUM VI Benchmark for Evaluating Visual-Inertial Odometry. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1680–1687, October 2018. arXiv: 1804.06120.

[110] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[111] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[112] G. Silveira, E. Malis, and P. Rives. An Efficient Direct Approach to Visual SLAM. *IEEE Transactions on Robotics*, 24(5):969–979, October 2008.

[113] Mike Smith, Ian Baldwin, Winston Churchill, Rohan Paul, and Paul Newman. The New College Vision and Laser Data Set. *The International Journal of Robotics Research*, 28(5):595–599, May 2009.

[114] Hauke Strasdat, J M M Montiel, and Andrew J Davison. Scale Drift-Aware Large Scale Monocular SLAM. page 8.

[115] Jrgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580, Vilamoura-Algarve, Portugal, October 2012. IEEE.

[116] Norhayati Mohd Suaib, Mohammad Hamiruce Marhaban, M Iqbal Saripan, and Siti Anom Ahmad. Performance evaluation of feature detection and feature matching for stereo visual odometry using sift and surf. In *2014 IEEE REGION 10 SYMPOSIUM*, pages 200–203. IEEE, 2014.

[117] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[118] A. Talukder, S. Goldberg, L. Matthies, and A. Ansar. Real-time detection of moving objects in a dynamic scene from moving robotic vehicles. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 2, pages 1308–1313 vol.2, October 2003.

[119] Jiexiong Tang, John Folkesson, and Patric Jensfelt. Geometric Correspondence Network for Camera Motion Estimation. *IEEE Robotics and Automation Letters*, 3(2):1010–1017, April 2018.

[120] Bernardo Teixeira, Hugo Silva, Anibal Matos, and Eduardo Silva. Deep Learning for Underwater Visual Odometry Estimation. *IEEE Access*, 8:44687–44701, 2020.

[121] E Todt and C Torras. Visual attention for the defection of natural outdoor landmarks. In *INTERNATIONAL JOURNAL OF PSYCHOLOGY*, volume 35, pages 236–236. PSYCHOLOGY PRESS 27 CHURCH RD, HOVE BN3 2FA, EAST SUSSEX, ENGLAND, 2000.

[122] Anne M Treisman and Garry Gelade. A feature-integration theory of attention. *Cognitive psychology*, 12(1):97–136, 1980.

[123] Mehmet Turan, Yasin Almalioglu, Helder Araujo, Ender Konukoglu, and Metin Sitti. Deep EndoVO: A Recurrent Convolutional Neural Network (RCNN) based Visual Odometry Approach for Endoscopic Capsule Robots. *Neurocomputing*, 275:1861–1870, January 2018. arXiv: 1708.06822.

[124] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380, April 1991.

[125] Abhinav Valada, Noha Radwan, and Wolfram Burgard. Deep Auxiliary Learning for Visual Localization and Odometry. *arXiv:1803.03642 [cs]*, March 2018. arXiv: 1803.03642.

[126] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2050, May 2017. arXiv: 1709.08429.

[127] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks. *The International Journal of Robotics Research*, 37(4-5):513–542, April 2018.

[128] Wei Wang, Muhamad Risqi U. Saputra, Peijun Zhao, Pedro Gusmao, Bo Yang, Changhao Chen, Andrew Markham, and Niki Trigoni. DeepPCO: End-to-End Point Cloud Odometry through Deep Parallel Neural Network. *arXiv:1910.11088 [cs]*, March 2020. arXiv: 1910.11088.

[129] Xiangwei Wang, Daniel Maturana, Shichao Yang, Wenshan Wang, Qijun Chen, and Sebastian Scherer. Improving Learning-based Ego-motion Estimation with Homomorphism-based Losses and Drift Correction. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 970–976, Macau, China, November 2019. IEEE.

[130] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[131] Fei Xue, Qiuyuan Wang, Xin Wang, Wei Dong, Junqiu Wang, and Hongbin Zha. Guided Feature Selection for Deep Visual Odometry. *arXiv:1811.09935 [cs]*, November 2018. arXiv: 1811.09935.

[132] Fei Xue, Xin Wang, Shunkai Li, Qiuyuan Wang, Junqiu Wang, and Hongbin Zha. Beyond Tracking: Selecting Memory and Refining Poses for Deep Visual Odometry. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8567–8575, Long Beach, CA, USA, June 2019. IEEE.

[133] Fei Xue, Xin Wang, Junqiu Wang, and Hongbin Zha. Deep visual odometry with adaptive memory. *arXiv preprint arXiv:2008.01655*, 2020.

[134] Mingsu Yan, Chaoxia Shi, and Yanqing Wang. A Monocular Visual Odometry Combining Edge Enhance with Deep Learning. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 370–374, Dali, China, December 2019. IEEE.

[135] Xiaochuan Yin, Xiangwei Wang, Xiaoguo Du, and Qijun Chen. Scale Recovery for Monocular Visual Odometry Using Depth Estimated with Deep Convolutional Neural Fields. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5871–5879, Venice, October 2017. IEEE.

[136] Khalid Yousif, Alireza Bab-Hadiashar, and Reza Hoseinnezhad. An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics. *Intelligent Industrial Systems*, 1(4):289–311, December 2015.

[137] Cheng Zhao, Li Sun, Pulak Purkait, Tom Duckett, and Rustam Stolkin. Learning monocular visual odometry with dense 3D mapping from dense 3D flow. *arXiv:1803.02286 [cs]*, July 2018. arXiv: 1803.02286.

[138] Huizhong Zhou, Benjamin Ummenhofer, and Thomas Brox. DeepTAM: Deep Tracking and Mapping. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, volume 11220, pages 851–868. Springer International Publishing, Cham, 2018.

[139] Michael Zollhöfer, Patrick Stotko, Andreas Görlitz, Christian Theobalt, Matthias Nießner, Reinhard Klein, and Andreas Kolb. State of the art on 3d reconstruction with rgb-d cameras. In *Computer graphics forum*, volume 37, pages 625–652. Wiley Online Library, 2018.

Appendix A

Architecture Configuration

This appendix will present the configuration and parameters used for training the model and executing the experiments. We present the configurations for the architectures used in the Pixel, City, and visual odometry experiments.

Network/Layer	**Neurons (in, out)**	**Activation**
glimpse.fc_xt_1	192, 192	relu
glimpse.fc_xt_2	192, 128	relu
glimpse.fc_xt_3	128, 128	linear
glimpse.fc_lt_1	2, 128	relu
glimpse.fc_lt_2	128, 128	linear
core.fc_gt_1	256, 256	relu
core.fc_gt_2	256, 256	linear
core.fc_ht_1	256, 256	relu
core.fc_ht_2	256, 256	linear
regressor.fc_rt_1	256, 128	relu
regressor.fc_rt_2	128, 2	tanh
locator.fc_mu_1	256, 64	relu
locator.fc_mu_2	64, 32	relu
locator.fc_mu_3	32, 2	tanh
locator.fc_std_1	256, 64	relu
locator.fc_std_2	64, 32	relu
locator.fc_std_3	32, 2	linear
baseliner.fc_bt_1	256, 64	relu
baseliner.fc_bt_2	64, 32	relu
baseliner.fc_bt_3	32, 1	linear

Table A.1: The model configuration for the experiments in the Pixel dataset.

Network/Layer	**Configuration (in, out, others)**	**Activation**
glimpse.conv_xt_1_1	1, 4, kernel_size=5, stride=1	leaky relu
glimpse.conv_xt_1_2	4, 8, kernel_size=5, stride=1	leaky relu
glimpse.conv_xt_2_1	1, 4, kernel_size=5, stride=1	leaky relu
glimpse.conv_xt_2_2	4, 8, kernel_size=5, stride=1	leaky relu
glimpse.conv_xt_3_1	1, 4, kernel_size=5, stride=1	leaky relu
glimpse.conv_xt_3_2	4, 8, kernel_size=5, stride=1	leaky relu
glimpse.fc_xt_1	1352, 64	linear
glimpse.fc_xt_2	1352, 32	linear
glimpse.fc_xt_3	1352, 32	linear
glimpse.fc_lt_1	2, 128	relu
glimpse.fc_lt_2	128, 128	linear
core.lstm_ht	256, 512	relu
regressor.fc_rt_1	512, 256	relu
regressor.fc_rt_2	256, 2	tanh
locator.fc_mu_1	512, 64	relu
locator.fc_mu_2	64, 32	relu
locator.fc_mu_3	32, 2	tanh
locator.fc_std_1	512, 64	relu
locator.fc_std_2	64, 32	relu
locator.fc_std_3	32, 2	linear
baseliner.fc_bt_1	512, 64	relu
baseliner.fc_bt_2	64, 32	relu
baseliner.fc_bt_3	32, 1	linear

Table A.2: The model configuration for the experiments in the City dataset.

Network/Layer	Configuration (in, out, others)	Activation
glimpse.conv_1_1	2, 32, kernel_size=3, stride=1	leaky relu
glimpse.conv_1_2	32, 32, kernel_size=3, stride=1	leaky relu
glimpse.conv_1_3	32, 64, kernel_size=3, stride=2	leaky relu
glimpse.conv_1_4	64, 64, kernel_size=3, stride=1	leaky relu
glimpse.conv_1_5	64, 128, kernel_size=3, stride=2	leaky relu
glimpse.conv_1_6	128, 128, kernel_size=3, stride=2	leaky relu
glimpse.conv_2_1	2, 32, kernel_size=5, stride=1	leaky relu
glimpse.conv_2_2	32, 32, kernel_size=5, stride=2	leaky relu
glimpse.conv_2_3	32, 64, kernel_size=5, stride=2	leaky relu
glimpse.conv_2_4	64, 64, kernel_size=5, stride=2	leaky relu
glimpse.conv_3_1	2, 32, kernel_size=5, stride=1	leaky relu
glimpse.conv_3_2	32, 32, kernel_size=5, stride=2	leaky relu
glimpse.conv_3_3	32, 64, kernel_size=5, stride=2	leaky relu
glimpse.conv_3_4	64, 64, kernel_size=5, stride=2	leaky relu
glimpse.fc_xt_1	2048, 256	linear
glimpse.fc_xt_2	1024, 128	linear
glimpse.fc_xt_3	1024, 128	linear
glimpse.fc_lt_1	2, 256	leaky relu
glimpse.fc_lt_2	256, 512	linear
core.lstm_ht_1	512, 1024	leaky relu
core.lstm_ht_2	1024, 1024	leaky relu
regressor.fc	1024, 256	leaky relu
regressor.fc_lt_1	256, 32	leaky relu
regressor.fc_lt_2	32, 2	linear
regressor.fc_lr_1	256, 32	leaky relu
regressor.fc_lr_2	32, 2	linear
locator.fc_mu_1	1024, 256	tanh
locator.fc_mu_2	256, 32	tanh
locator.fc_mu_3	32, 2	tanh
locator.fc_std_1	1024, 256	relu
locator.fc_std_2	256, 32	relu
locator.fc_std_3	32, 2	linear
baseliner.fc_bt_1	1024, 256	tanh
baseliner.fc_bt_2	256, 32	tanh
baseliner.fc_bt_3	32, 1	linear

Table A.3: The baseline RAM-VO configuration for the experiments in the KITTI dataset.

Network/Layer	Configuration (in, out)	Activation
actor.fc_1	1024, 128	tanh
actor.fc_2	128, 32	tanh
actor.fc_3	32, 2	tanh
critic.fc_1	1024, 128	tanh
critic.fc_2	128, 32	tanh
critic.fc_3	32, 1	linear

Table A.4: The PPO network replaced the locator and baseliner networks on the baseline RAM-VO.

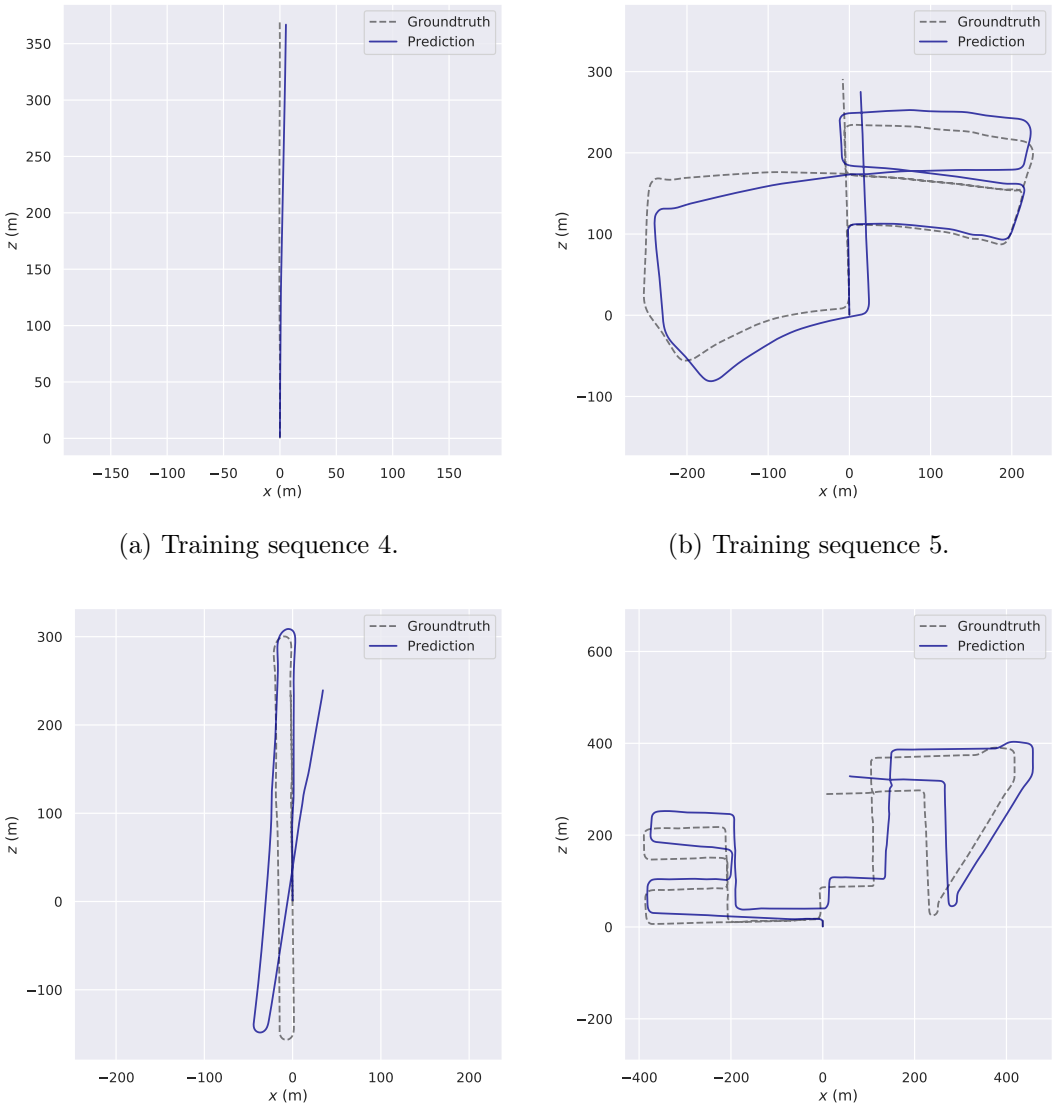
Network/Layer	Configuration (in, out, others)	Activation
context.conv_1	2, 16, kernel_size=5, stride=1	leaky relu
context.conv_2	16, 16, kernel_size=3, stride=2	leaky relu
context.conv_3	16, 32, kernel_size=3, stride=2	leaky relu
context.conv_4	32, 32, kernel_size=3, stride=2	leaky relu
context.fc_xt	6656, 1024	leaky relu

Table A.5: The context network added to the baseline RAM-VO.

Appendix B

Trajectory Predictions and Metrics

This appendix will present the trajectories' predictions and metrics.



(a) Training sequence 4.

(b) Training sequence 5.

(c) Training sequence 6.

(d) Training sequence 8.

Figure B.1: Other trajectory's prediction for the baseline RAM-VO on the train set.

	Train set		
Seq.	$t_{rpe}(\%)$	$r_{rpe}(°)$	ATE(m)
0	1.0763	0.4401	4.6062
2	1.1153	0.3974	6.1720
4	0.4678	0.3621	0.2870
5	0.9404	0.3878	3.7622
6	1.1564	0.4126	1.5388
8	0.9798	0.3978	2.9054
9	1.1526	0.4560	3.2415
Mean	**0.9841**	**0.4077**	**3.2162**
	Test set		
3	15.6642	5.3904	24.6669
7	14.5508	9.3104	28.7291
10	18.8916	8.2986	54.9151
Mean	**16.3689**	**7.6665**	**36.1037**

(a) 1 glimpse at center

	Train set		
Seq.	$t_{rpe}(\%)$	$r_{rpe}(°)$	ATE(m)
0	23.1781	8.1843	305.9996
2	17.8436	5.3696	246.9051
4	11.5616	1.0198	12.4498
5	17.3727	6.2303	103.0818
6	11.8251	3.6210	31.7991
8	17.6408	4.5736	114.9046
9	16.1914	5.5823	74.9670
Mean	**16.5162**	**4.9401**	**127.1582**
	Test set		
3	19.6587	6.1033	24.9080
7	28.9232	10.4805	49.0730
10	29.3264	7.2343	52.6815
Mean	**25.9694**	**7.9394**	**42.2208**

(b) 1 glimpse random

	Train set		
Seq.	$t_{rpe}(\%)$	$r_{rpe}(°)$	ATE(m)
0	6.8526	2.8410	110.9812
2	3.7823	1.4547	70.0347
4	0.8177	1.2061	0.6088
5	3.8986	1.7768	21.6607
6	2.1190	0.9397	4.7911
8	4.7732	1.7542	36.2751
9	2.8990	1.2459	10.8877
Mean	**3.5918**	**1.6027**	**36.4627**
	Test set		
3	11.0282	5.1012	7.6712
7	7.4283	3.6165	12.8483
10	14.3307	3.2376	31.7342
Mean	**10.9291**	**3.9851**	**17.4179**

(c) 4 glimpses

	Train set		
Seq.	$t_{rpe}(\%)$	$r_{rpe}(°)$	ATE(m)
0	3.8358	1.7240	34.7433
2	3.4659	1.3217	46.6921
4	1.1637	0.6898	1.0144
5	3.6743	1.7049	15.9267
6	2.9833	1.5528	9.4757
8	3.3647	1.3694	21.4095
9	2.6575	1.3874	12.9188
Mean	**3.0208**	**1.3928**	**20.3115**
	Test set		
3	10.3137	4.8420	10.1368
7	7.2814	3.6886	13.9782
10	15.0694	4.0881	35.3034
Mean	**10.8881**	**4.2062**	**19.8061**

(d) 8 glimpses

	Train set		
Seq.	$t_{rpe}(\%)$	$r_{rpe}(°)$	ATE(m)
0	7.4362	3.0950	136.6553
2	5.8078	2.1137	129.7836
4	1.3799	0.7238	1.3612
5	6.5283	3.0681	37.8829
6	2.9707	1.1420	5.3964
8	6.3407	2.5983	56.5116
9	6.1263	2.1442	39.7590
Mean	**5.2271**	**2.1264**	**58.1929**
	Test set		
3	11.5491	4.4609	8.0572
7	9.8063	5.1468	20.2515
10	16.0270	2.7750	34.7048
Mean	**12.4608**	**4.1275**	**21.0045**

(e) 8 glimpses random

	Train set		
Seq.	$t_{rpe}(\%)$	$r_{rpe}(°)$	ATE(m)
0	4.6368	2.0435	68.8990
2	3.3005	1.3302	67.0351
4	0.4292	0.7193	0.2891
5	4.2827	1.9339	24.2714
6	2.6755	1.2114	7.7146
8	4.6252	1.9255	41.7535
9	3.3946	1.3636	17.6544
Mean	**3.3349**	**1.5039**	**32.5167**
	Test set		
3	13.0198	5.9033	15.9198
7	7.9654	5.0906	15.1237
10	18.5594	6.3199	43.2433
Mean	**13.1815**	**5.7713**	**24.7622**

(f) 12 glimpses

Table B.1: RPE and ATE metrics by sequence for the baseline RAM-VO.

	Train set		
Seq.	$t_{rpe}(\%)$	$r_{rpe}(°)$	ATE(m)
0	3.3424	1.6220	29.2420
2	4.8571	1.7879	70.5255
4	1.0508	0.5456	0.9110
5	4.8045	1.9640	29.7684
6	2.1752	1.0551	5.8504
8	4.9200	1.9146	23.6712
9	5.9543	1.9397	36.0784
Mean	**3.8720**	**1.5470**	**28.0067**
	Test set		
3	5.7210	3.0849	4.4624
7	9.1726	5.6282	12.6231
10	13.8508	3.2444	27.0254
Mean	**9.5815**	**3.9858**	**14.7036**

(a) 1024 hidden units

	Train set		
Seq.	$t_{rpe}(\%)$	$r_{rpe}(°)$	ATE(m)
0	4.6124	1.9951	50.8312
2	6.5830	2.3823	111.2751
4	3.5543	2.1643	1.6722
5	6.2361	2.4078	33.8585
6	3.2783	1.3283	5.9350
8	5.7592	2.2392	39.1822
9	6.5545	2.4480	43.5041
Mean	**5.2254**	**2.1378**	**40.8940**
	Test set		
3	16.3040	7.7288	16.1894
7	9.2066	7.7197	21.7765
10	16.0297	4.4694	26.9866
Mean	**13.8468**	**6.6393**	**21.6508**

(b) 512 hidden units

	Train set		
Seq.	$t_{rpe}(\%)$	$r_{rpe}(°)$	ATE(m)
0	6.1718	2.5393	52.4030
2	5.1550	1.9580	54.9758
4	1.7816	1.7199	1.2154
5	5.8573	2.7780	22.5703
6	8.1119	2.9400	23.4548
8	6.7982	2.4358	26.2370
9	3.6509	1.6576	15.0408
Mean	**5.3610**	**2.2898**	**27.9853**
	Test set		
3	7.0859	4.0115	6.7140
7	7.5475	4.3027	11.6459
10	15.0185	5.1179	30.0463
Mean	**9.8839**	**4.4774**	**16.1354**

(c) 256 hidden units

Table B.2: RPE and ATE metrics by sequence for the RAM-VO with PPO.

	Train set		
Seq.	$t_{rpe}(\%)$	$r_{rpe}(°)$	ATE(m)
0	3.3473	1.4112	45.1485
2	2.6061	0.9661	23.6524
4	0.8735	1.2657	0.5595
5	4.0874	1.6830	25.5541
6	1.8174	0.7626	4.0826
8	2.6959	1.1845	18.7020
9	2.3593	0.8943	3.9882
Mean	**2.5410**	**1.1668**	**17.3839**
	Test set		
3	8.8108	5.3063	9.2007
7	10.0028	6.1038	22.4449
10	16.4538	4.5277	39.4764
Mean	**11.7558**	**5.3126**	**23.7073**

Table B.3: RPE and ATE metrics by sequence for the RAM-VO with context.

	Train set		
Seq.	$t_{rpe}(\%)$	$r_{rpe}(°)$	ATE(m)
0	5.9431	2.4440	73.9850
2	7.3894	2.6324	149.6708
4	1.5093	1.0244	0.6390
5	7.6114	3.3830	42.1866
6	4.1802	1.7005	6.7460
8	7.8974	3.1604	69.9174
9	7.6905	2.8151	54.6501
Mean	**6.0316**	**2.4514**	**56.8278**
	Test set		
3	16.0100	9.6344	19.0475
7	9.2246	5.8308	20.1384
10	19.9846	6.8716	41.2782
Mean	**15.0731**	**7.4456**	**26.8214**

(a) 4 frames on the CNN channels

	Train set		
Seq.	$t_{rpe}(\%)$	$r_{rpe}(°)$	ATE(m)
0	4.0976	1.6976	18.5851
2	3.8759	1.5965	35.9077
4	0.8953	1.6918	0.6451
5	4.5682	1.9319	19.3594
6	3.4979	1.4174	4.5701
8	3.7940	1.4561	18.4065
9	3.7830	1.4934	11.9661
Mean	**3.5017**	**1.6121**	**15.6343**
	Test set		
3	13.3778	6.6116	15.8119
7	8.2596	4.7371	14.5764
10	16.4871	5.1957	36.9847
Mean	**12.7082**	**5.5148**	**22.4577**

(b) 2-2 frames sequential

Table B.4: RPE and ATE metrics by sequence for the RAM-VO with sequential information.