RAM-VO: A Recurrent Attentional Model for Visual Odometry

Iury Cleveston[1], **Esther L. Colombini**[1]

[1]Laboratory of Robotics and Cognitive Systems (LaRoCS)
Institute of Computing, University of Campinas
Campinas, São Paulo, Brazil

{iury.cleveston,esther}@ic.unicamp.br

Abstract. *Determining the agent's pose is fundamental for developing autonomous vehicles. Visual Odometry (VO) algorithms estimate the egomotion using only visual differences from the input frames. The most recent VO methods implement deep-learning techniques using convolutional neural networks (CNN) widely, adding a high cost to process large images. Also, more data does not imply a better prediction, and the network may have to filter out useless information. In this context, we incrementally formulate a lightweight model called RAM-VO to perform visual odometry regressions using large monocular images. Our model is extended from the Recurrent Attention Model (RAM), which has emerged as a unique architecture that implements a hard attentional mechanism guided by reinforcement learning to select the essential input information. Our methodology modifies the RAM and improves the visual and temporal representation of information, generating the intermediary RAM-R and RAM-RC architectures. Also, we include the optical flow as contextual information for initializing the RL agent and implement the Proximal Policy Optimization (PPO) algorithm to learn a robust policy. The experimental results indicate that RAM-VO can perform regressions with six degrees of freedom using approximately 3 million parameters. Additionally, experiments on the KITTI dataset confirm that RAM-VO produces competitive results using only 5.7% of the input image.*

M.Sc. Dissertation submitted to CTDR 2021 - Defense date: 05/07/21

Supervisor: Prof. Dr. Esther Luna Colombini

1. Introduction

Autonomous vehicles have been a long-standing research topic for the scientific community. Building vehicles capable of functioning without human supervision is challenging, requiring the vehicle's pose determination. Visual Odometry (VO) is the field concerned with estimating the egomotion of an agent using only visual differences from the input frames. 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 [Scaramuzza and Fraundorfer 2011]. Direct and indirect methods have been developed to estimate the agent's pose by capturing the apparent motion from sequential frames. These methods require the environment to have satisfactory illumination, texture, and the subsequent frames must overlap.

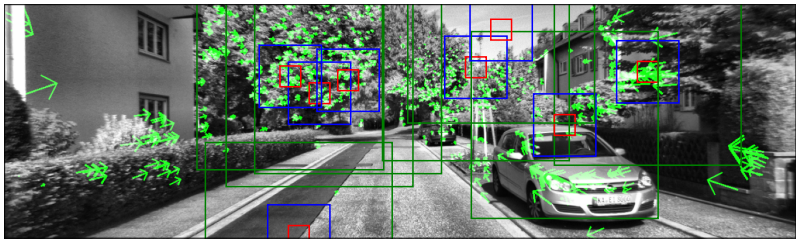

Figure 1. Sequences of eight observations defined by a learned policy with the sparse optical flow.

However, hand-crafted solutions based on either direct or indirect approaches become complicated due to the problem's nature, which presents a vast search space, and considerable non-linearities. Usually, these solutions do not cover all possibilities and suffer from robustness issues when used in different environments. In recent years, Deep Learning (DL) techniques have appeared as a novel way to perform statistical learning on real data captured from the environment, showing promising results in complex datasets, such as KITTI [Geiger et al. 2013]. 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 multiple nonlinear factors that impact the scene generation and motion [Goodfellow et al. 2016].

Although DL provides significant advantages, the visual odometry field makes unrestricted 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, and the network may have to filter out useless information. Therefore, the implementation of lightweight architectures 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. Particularly, attention 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) [Mnih et al. 2014] has emerged as a novel architecture that implements a recurrent hard attentional mechanism to incrementally select the essential pieces of information. The attentional mechanism is guided by a REINFORCE [Williams 1992] policy learned via reinforcement learning (RL). Despite this, RAM was introduced essentially as a concept proof, only implemented for classification tasks on the MNIST dataset, not possessing the complex structures to deal with visual odometry tasks.

Therefore, this work proposes constructing a monocular end-to-end visual odometry architecture – RAM-VO – employing the reinforcement learning paradigm to train a hard attentional glimpse sensor over time. In this sense, we incrementally extended the RAM architecture by implementing spatial and temporal structures to operate with complex visual inputs; our methodology establishes the RAM-R and RAM-RC as intermediary models. In the end, the optical flow is added to provide contextual information to initialize the RL agent, and the Proximal Policy Optimization (PPO) [Schulman et al. 2017] enables the learning of robust policies. RAM-VO can predict 6-degree-of-freedom (DoF) poses in real-world sequences and is more computationally efficient than similar methods. 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. Contributions

This work provides the following contributions:

- A lightweight VO method that selects the important input information via attentional mechanisms;
- The first visual odometry architecture that implements reinforcement learning in part of the pipeline;
- Several experiments on KITTI [Geiger et al. 2013] sequences demonstrating the validity and efficiency of RAM-VO;
- A Survey of Deep Supervised Learning for Visual Odometry, to be submitted soon to the Journal of Neurocomputing;
- RAM-VO: Less is more in Visual Odometry, submitted to IEEE Robotics and Automation Letters (RA-L).

2. Related Work

The first visual odometry method implemented with deep learning was proposed by Konda et al. [Konda and Memisevic 2015] in 2015. Their model implemented an end-to-end CNN architecture to estimate direction and velocity from raw stereo images; however, the problem was formulated as a classification task, and the experimental results were not satisfactory due to limited data. After that, the architectures started to coupled LSTM layers to provide temporal representation. In 2017, Wang et al. [Wang et al. 2017] proposed the DeepVO, which is an end-to-end monocular architecture capable of extracting visual features directly from the input raw images with a CNN and determining temporal relation with an LSTM.

At the same time, Flowdometry [Muller and Savakis 2017], based on the FlowNet architecture, extracted the optical flow from the input images. The pipeline's first stage computes the optical flow from monocular images using the FlowNetS, while the second stage computes incremental changes in angle and displacements. Also, Zhao et al. [Zhao et al. 2018] 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, and the FlowNet computes the optical flow, while the DepthNet computes the depth map. Further, the DPC-Net [Peretroukhin and Kelly 2018] model aims to integrate the representation capabilities of deep neural networks with the efficiency of geometric and probabilistic algorithms. DPC-Net implements a CNN-based architecture to learns the corrections for the pose estimator. Yin et al. [Yin et al. 2017] proposed an architecture able to recover the scale by estimating the depth from the input monocular images based on a modified ResNet and conditional random fields. Tang et al. [Tang et al. 2018] proposes a Geometric Correspondence Network, consisting of a CNN trained together with an RNN to detect the keypoints' location and generate descriptors.

Several other methods provide technological contributions. The most recent employ deep learning structures and demand the adoption of large and representative datasets. Conversely, lightweight and efficient methods are essential to the field. Reinforcement Learning (RL) and attention applied to visual odometry can provide a solution in such scenarios. The model becomes efficient by selecting only the necessary input data,

and learning a robust policy can reduce the drift error. However, we have not found any method that implements RL in any part of the visual odometry pipeline.

2.1. The Recurrent Attention Model (RAM)

The Recurrent Attention Model (RAM)[Mnih et al. 2014] implements a hard attention mechanism similar to the biological visual system, which progressively constructs an informative latent space by multiple observations in the input image. Hard attention requires the model to consider only the relevant elements while discarding the others completely [Correia and Colombini 2021]. The observations are iteratively integrated into the model, providing the knowledge to perform the classification. The location of each observation is determined by a policy learned through REINFORCE [Williams 1992]. RAM is constituted of four different networks, as shown in Figure 2.

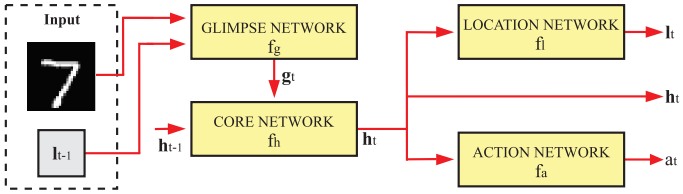

Figure 2. The RAM architecture is constituted of four different networks.

The glimpse network f_g represents the attentional system and comprises a glimpse sensor to extract meaningful patches from the input images. First, the glimpse sensor receives an image $\mathbf{x}_t$ and a location $\mathbf{l}_{t-1}$ as input. Then, several patches are extracted in different resolutions, centered at the location $\mathbf{l}_{t-1}$. This process builds a pyramidal-like structure $\rho(\mathbf{x}_t, \mathbf{l}_{t-1})$ representing what was observed on the image. Finally, the glimpse network concatenates ρ_t and $\mathbf{l}_{t-1}$ to include the location where the information was extracted, resulting in the final vector $\mathbf{g}_t$. The core network f_h stores the multiple observations by receiving the feature vector $\mathbf{g}_t$ and the previous internal state $\mathbf{h}_{t-1}$ as input at every time step t. Through fully connected layers, the core network outputs the current internal state $\mathbf{h}_t$, which condenses all the sequential information provided by the glimpse network. The location network f_l generates the location $\mathbf{l}_t$ for the subsequent observation by sampling a Gaussian distribution with two dimensions $(x,\ y)$ and a fixed standard deviation. For each subsequent observation, a novel Gaussian distribution is generated by using the internal state $\mathbf{h}_t$ to parameterize the mean μ_t. After all observations, the action network f_a consumes the internal state $\mathbf{h}_t$ to predict the class a_t, which is the final goal.

The hard attention mechanism introduces a non-differentiable structure and thus requires reinforcement learning to train the model. Therefore, the RL setup is an instance of a Partially Observable Markov Decision Process, in which the true state of the environment is unobserved. In this sense, the glimpse sensor is the agent, the whole image is the environment, and the rewards are defined according to the success in the classification. The goal is to maximize the return $G = \sum_{t=1}^{T} r_t$, which is sparse and delayed, via $J(\boldsymbol{\theta}) = \mathbb{E}_{p(\mathbf{h}_{1:T};\boldsymbol{\theta})}[G]$, where $p(\mathbf{h}_{1:T};\boldsymbol{\theta})$ depends on the policy. Maximizing $J(\boldsymbol{\theta})$ is not trivial because it involves an expectation about high-dimension iteration sequences. However, we can obtain an approximation of the gradient with the REINFORCE rule [Williams 1992] as $\nabla J(\boldsymbol{\theta}) = \frac{1}{M}\sum_{i=1}^{M}\sum_{t=1}^{T} \nabla \log\pi\left(\mathbf{l}_t^i|\mathbf{h}_{1:t}^i;\boldsymbol{\theta}\right)(G_t^i - b_t)$, where $\mathbf{h}_{1:t}^i$ are the sequences obtained by running the current policy π_θ for $i = 1, ..., M$ episodes, G_t^i is the accumulated reward after executing action $\mathbf{l}_t^i$, and b_t is the baseline value, which

reduces the variance for the gradient updates. Although bringing innovative ideas from biology, RAM was proposed mainly as a proof of concept, lacking the necessary complexity to deal with high-resolution images and regression tasks.

3. The RAM-VO Architecture

The original RAM architecture was created for simple classification tasks. In this sense, we have developed a methodology to extend it to visual odometry tasks by increasing the architecture's complexity to deal with more challenging input data. The methodology comprises three steps: a) creating the RAM-R to learn the displacement between single pixels in two different images; b) increasing the RAM-R complexity to provide the displacement between two images with complex visual structures, generating RAM-RC; c) producing the final RAM-VO for visual odometry tasks by learning the optical flow and improving the temporal representation, policy parametrization, and pose regression.

3.1. RAM-R: Simple Regression on Pixel Dataset

Adapting RAM to regression tasks requires two glimpse networks, allowing the architecture to consume two different images simultaneously. Therefore, RAM-R can find the same features in both images and determine their correspondence. To do so, we changed the original action 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. Unlike 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, in which 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. We evaluated our results using our own created dataset called Pixel, consisting of black images with a single white pixel, with a 100×100 pixels resolution. The RAM-R aimed to regress the distance (x, y) between pixels from two separate input images. The experimental results achieved an MAE of 3.118 in the test set, which we considered satisfactory to continue increasing the model complexity.

3.2. RAM-RC: Complex Regression on City Dataset

Once the RAM-R performed satisfactory regressions with a simple problem, we made the regression more complicated. The City dataset comprises images with 300×300 pixels extracted from a city's panoramic image in high resolution. This dataset provides an indispensable environment to understand the agent's behavior when dealing with highly nonlinear visual information. In this sense, we can observe how the agent uses the glimpse sensor to choose an action; and how the sequence of observations generates a usable latent space for predictions. RAM-RC included 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 proposed only one LSTM layer for this problem, although more layers allow the model to have a hierarchical representation of the sequential data. For this purpose, we modified the

glimpse network 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. The experimental results achieved an MAE of 3.810 in the test set.

3.3. 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. 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. We were inspired by the FlowNetS 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 (Figure 3).

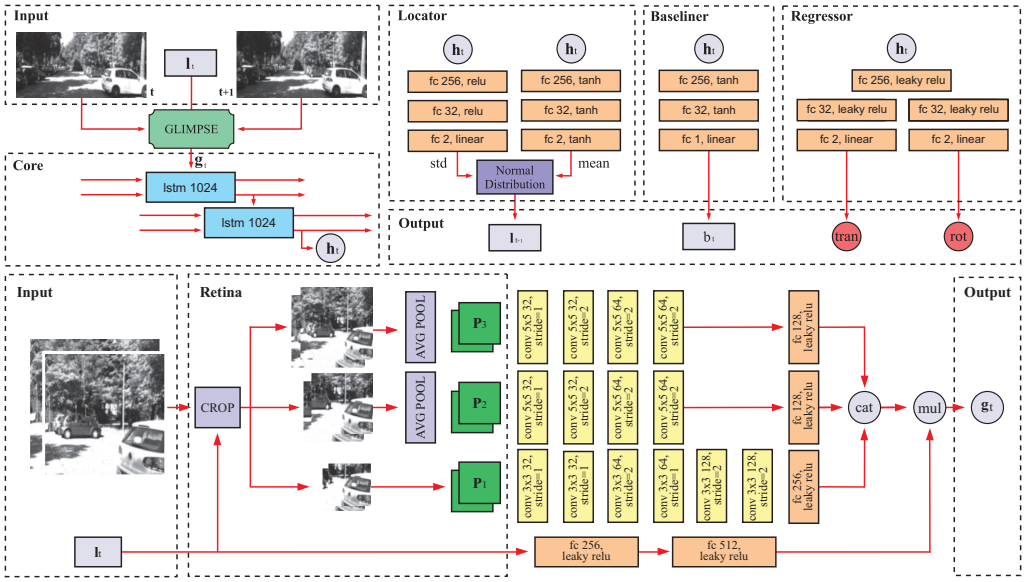

Figure 3. Overview of RAM-VO (top) and Glimpse Network (bottom).

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, applying the same filters 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. Hence, we maintained the convolution operations but used only six layers for the high-resolution scale, with 128 channels in the last operation. In this sense, we observed that

convolutional operations with smaller kernels significantly minimize the loss; therefore, we chose to keep the high-resolution scale with a kernel size of 3×3 pixels and the other scales with a 5×5 kernel since they represent large areas. We define padding values as zero, removing entirely pooling operations. The input information's dimensionality is reduced by varying the stride between 1 and 2 during the filters' application.

The supervised loss and the reward function are both defined in terms of the MSE. We also evaluated MAE 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 $L = \frac{1}{N}\sum_{i=1}^{N} \|\hat{\boldsymbol{p}} - \boldsymbol{p}\|_2^2 + k \|\hat{\boldsymbol{\varphi}} - \boldsymbol{\varphi}\|_2^2$, 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 $R = \frac{1}{1+L}$. 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 especially necessary when the ground-truth values are not normalized, and the orientation component must be compensated since they present a lower variation range. 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 $[\phi, \theta, \psi, x, y, z]^T$.

3.4. Learning a Policy with Proximal Policy Optimization (PPO)

The REINFORCE [Williams 1992] algorithm is known for presenting convergence issues and slowness; this occurs by abrupt updates on the policy's parameters, which can harm the entire training by converging to suboptimal regions. The Proximal Policy Optimization (PPO) [Schulman et al. 2017] intends to solve these issues by updating the policy inside trusted regions. The PPO's surrogate function determines that the current policy must be close to the last one, avoiding large parameters shifts. The use of memory replay also played an essential role since it allowed 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 essential 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 units.

3.5. Providing Optical Flow as Contextual Information

A further step was to initialize the last LSTM layer in the core network with contextual information to give the RL agent an image overview for subsequent observations. The most helpful information is the optical flow extracted between the two frames since it resumed the salient features required to predict the motion. We decided to inform the dense optical flow extracted by the Farneback method and use CNN layers to determine their importance, as shown in Figure 4. Initializing the RL agent with the dense optical helped to determine the most valuable image regions for further exploration through the subsequent glimpses. The optical flow provided information on whether the vehicle is either in rotational or translational motion so that the integration of the observations can

occur appropriately. The Farneback method generated an image pyramid with increasing resolutions, enabling tracking large object's motion; we kept the pyramid levels as 10 with a window size of 40x40 pixels. The motion was tracked from the lowest-resolution level and refined as the keypoints were propagated to the next level.

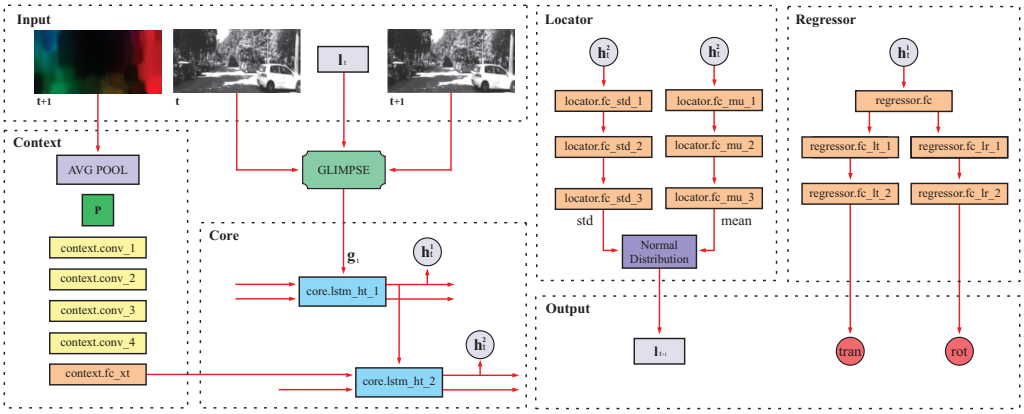

Figure 4. The RAM-VO architecture with contextual information.

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 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 did not employ pooling operations between layers. 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. Afterward, the agent captured 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 provided the 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 allowed the agent to have a scene overview of interesting regions and determine the first glimpse location instead of randomly chosen.

4. Materials and Methods

4.1. Dataset

In this work, we used the KITTI [Geiger et al. 2013] dataset for the final experiments, composed of the first eleven (00-10) sequences that have ground-truth information. Specifically, we used the grayscale images provided by the left camera, resized to 1200×360 pixels in resolution. To compare our results with other methods, we chose sequences 0, 2, 4, 5, 6, 8, 9 for training, sequences 10 for validation, and sequences 3, 7 for testing. We preprocessed each image by equalizing the pixel intensity histogram in small windows of 8×8 pixels with the Contrast Limited Adaptive Histogram Equalization method; then, we normalized the preprocessed images with the z-score function.

4.2. Evaluation Metrics

To evaluate our results, we computed the method's global consistency with the **Absolute Trajectory Error (ATE)**, given at instant i by $\mathcal{E}_i = \mathbf{G}_i^{-1}\mathbf{A}\mathbf{H}_i$, where $\mathbf{G}_i$ is the ground-truth pose, $\mathbf{H}_i$ is the estimated trajectory pose, $\mathbf{A}$ is the best alignment transformation. Also, we measured the trajectory's local consistency with the **Relative Pose Error (RPE)**, defined at instant i by $\mathcal{F}_i = \frac{\mathbf{H}_i^{-1}\mathbf{H}_{i+k}}{\mathbf{G}_i^{-1}\mathbf{G}_{i+k}}$, where k is a fixed time interval, which determines the consistency accuracy. We computed RPE by averaging all sub-sequences ranging from 100 to 800 meters in KITTI's sequences.

4.3. Hardware and Hyperparameters

We built this work in Python 3 with Pytorch. The hardware used for training the models was an Intel Core i7-10700KF @ 3.80GHz, Nvidia RTX 2060 with 6Gb, and Cuda v11.1. The model configuration consisted of 3 image patches of 32×32 pixels, batch size of 128, supervised learning rate of 1×10^{-4}, and RL learning rate of 1×10^{-6}; we employed the Adam optimizer for both networks. We trained the models for 400 epochs without early stopping or learning rate decay. The average training time was around 13 hours, and the inference time is 35ms for a pair of frames.

5. Experimental Results

We conducted several experiments (Table 1) to validate the number of observations on building a internal state $\mathbf{h}_t$ for visual odometry. Also, we altered the number of observations from 1 to 12 and also tested with random observations. Then, we replaced the REINFORCE algorithm with PPO to estimate the impact of the policy on the generalization; subsequently, we altered the internal state $\mathbf{h}_t$ capacity from 1024 to 256 hidden units to evaluate the impact on the drift error (Table 2).

5.1. The Impact of Observations

Experiments with 4, 8, and 12 observations have the locations determined by the learned policy, and the model achieved the best generalization results with 4 and 8 glimpses; 12 glimpses have not provided better results. More observations demand more from the core and locator networks since a single poor observation can harm the entire internal state

Table 1. The impact of the number of observation. $\bar{\mathrm{t}}_{\mathrm{rpe}}$ represents the average translational RMSE drift (%) on length of 100m to 800m. $\bar{\mathrm{r}}_{\mathrm{rpe}}$ is the average rotational RMSE drift (°/100m) on length of 100m to 800m. $\overline{\mathrm{ATE}}$ represents the average absolute trajectory error.

	Train set			Test set		
Configuration	$\bar{\mathrm{t}}_{\mathrm{rpe}}$	$\bar{\mathrm{r}}_{\mathrm{rpe}}$	$\overline{\mathrm{ATE}}$	$\bar{\mathrm{t}}_{\mathrm{rpe}}$	$\bar{\mathrm{r}}_{\mathrm{rpe}}$	$\overline{\mathrm{ATE}}$
1 glimpse at center	0.984	0.408	3.216	16.369	7.666	36.104
1 glimpse random	16.516	4.940	127.158	25.969	7.939	42.221
4 glimpses	3.599	1.608	36.463	10.929	**3.985**	**17.418**
8 glimpses	**3.021**	**1.393**	**20.311**	**10.888**	4.206	19.806
8 glimpses random	5.227	2.126	58.193	12.461	4.127	21.004
12 glimpses	3.335	1.504	32.517	13.181	5.771	24.762

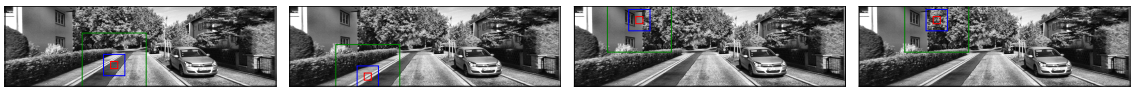

Figure 5. The sequence of four observations on the first frame on sequence 2.

and delay, even more, the sparse reward. Considering the experiment with 8 observations, the totality of input information is 5.7% of the total available. We conclude that the agent is retrieving high informative regions for most observations, and the learned policy indicates a preference for observing the left-center part of the image, as shown in Figure 5. Single observations have not provided enough information for building effective models. Observing the image's center harmed the generalization by reducing the data diversity necessary to learn complex behaviors. Further, a single random observation impeded even the learning by capturing little and sparse information. Good predictions with random observations require learning the general dynamics since the input space is ample and the model's capacity is limited.

5.2. The Impact of Capacity and PPO

Replacing REINFORCE algorithm with PPO and reducing the internal state $\mathbf{h}_t$ capacity from 1024 to 256 hidden units enabled us to know the minimum capacity required to deliver robust and lightweight models. Therefore, we made 8 observations and computed the statistics for three distinct executions. PPO with 1024 hidden units provided the best generalization, and decreasing the number of parameters strongly affects generalization on average. Although PPO 1024 had the best performance, the difference in the results may not justify a three-fold increase in parameters. PPO algorithm is susceptible to initialization; hence, we selected the best models of three executions to predict the trajectories (Figure 6). PPO 256 has only 2.92 million parameters and provides results similar to PPO 1024 on the best execution. Additionally, we confirmed that the PPO provided a slightly better generalization capacity than the REINFORCE.

Table 2. The impact of PPO and the internal state capacity.

		Train set		Test set	
Config.	**Param.**	$\bar{t}_{rpe}$	$\bar{r}_{rpe}$	$\bar{t}_{rpe}$	$\bar{r}_{rpe}$
PPO 1024	16.54M	**4.09** $\pm$ 0.27	**1.68** $\pm$ 0.17	**10.63** $\pm$ 1.38	**4.37** $\pm$ 0.43
PPO 512	5.84M	7.07 $\pm$ 2.43	2.73 $\pm$ 0.74	15.49 $\pm$ 4.06	7.08 $\pm$ 1.87
PPO 256	2.92M	7.29 $\pm$ 1.68	2.79 $\pm$ 0.44	15.50 $\pm$ 5.18	6.68 $\pm$ 2.00

5.3. Comparison with Literature

We compared our proposed model RAM-VO using PPO with traditional VO methods in literature such as ORB-SLAM [Mur-Artal et al. 2015], DeepVO [Wang et al. 2017], and ESP-VO [Wang et al. 2018], as shown in Table 3. 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. RAM-VO obtained competitive results using less input information, around 5.7% of the total available. 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. DeepVO and ESP-VO are based on the FlowNet architecture and have

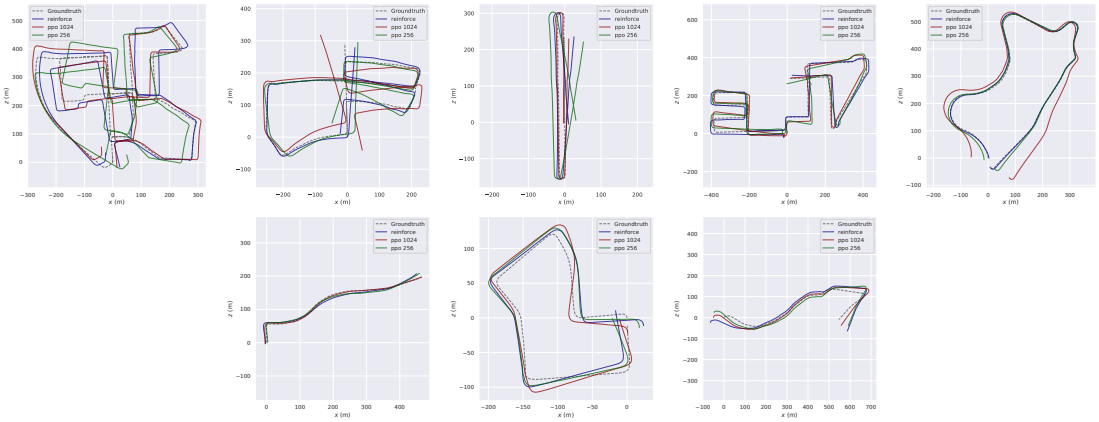

Figure 6. RAM-VO trajectories' predictions for training (first row) and testing (second row) sequences on the KITTI dataset using our best model.

more convolutional layers and channels than RAM-VO. 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. Concurrent methods regularly pass 32 million parameters, especially when they are extensions of architectures like AlexNet, and FlowNet.

Table 3. RAM-VO results compared to other methods on test sequences.

Method	Data %	Seq. 03		Seq. 07		Seq. 10	
		t_{rpe}	r_{rpe}	t_{rpe}	r_{rpe}	t_{rpe}	r_{rpe}
ORB-SLAM	-	21.07	18.36	24.53	38.90	86.51	98.90
DeepVO	100	8.49	6.89	3.91	4.60	**8.11**	8.83
ESP-VO	100	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

6. Conclusion

In this work, we incrementally created the RAM-VO architecture for end-to-end visual odometry using monocular images. RAM-VO extends the RAM [Mnih et al. 2014] for complex regression tasks while also defining an attentional glimpse sensor for high-resolution input images. The observations are guided by reinforcement learning and recurrently integrated for subsequent use. RAM-VO is the first architecture for visual odometry that implemented reinforcement learning to the best of our knowledge. Experimental results indicate that RAM-VO could regress 6-DoF poses in the KITTI dataset with generalization and utilizing around 5.7% of the total input data. Also, we provided alternative models with the optical flow for a bottom-up attentional system and PPO for learning more beneficial policies.

Acknowledgment

The authors would like to thank the Brazilian National Council for Scientific and Technological Development (CNPq), grant 130834/2019-0, and Bradesco Bank for supporting our research.

References

Correia, A. d. S. and Colombini, E. L. (2021). Attention, please! a survey of neural attention models in deep learning. *arXiv preprint arXiv:2103.16775*.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. *INT J ROBOT RES*, 32(11):1231–1237.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts.

Konda, K. and Memisevic, R. (2015). Learning Visual Odometry with a Convolutional Network:. In *Proceedings of the 10th ICCV Theory and Applications*, pages 486–490, Berlin, Germany. SCITEPRESS - Science and and Technology Publications.

Mnih, V., Heess, N., Graves, A., and Kavukcuoglu, K. (2014). Recurrent models of visual attention. *arXiv preprint arXiv:1406.6247*.

Muller, P. and Savakis, A. (2017). Flowdometry: An Optical Flow and Deep Learning Based Approach to Visual Odometry. In *2017 IEEE WACV*, pages 624–631, Santa Rosa, CA, USA. IEEE.

Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. (2015). Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163.

Peretroukhin, V. and Kelly, J. (2018). DPC-Net: Deep Pose Correction for Visual Localization. *IEEE Robotics and Automation Letters*, 3(3):2424–2431.

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

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Tang, J., Folkesson, J., and Jensfelt, P. (2018). Geometric Correspondence Network for Camera Motion Estimation. *IEEE Robotics and Automation Letters*, 3(2):1010–1017.

Wang, S., Clark, R., Wen, H., and Trigoni, N. (2017). DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks. *2017 IEEE ICRA*, pages 2043–2050.

Wang, S., Clark, R., Wen, H., and Trigoni, N. (2018). End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks. *INT J ROBOT RES*, 37(4-5):513–542.

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

Yin, X., Wang, X., Du, X., and Chen, Q. (2017). Scale Recovery for Monocular Visual Odometry Using Depth Estimated with Deep Convolutional Neural Fields. In *2017 IEEE ICCV*, pages 5871–5879, Venice. IEEE.

Zhao, C., Sun, L., Purkait, P., Duckett, T., and Stolkin, R. (2018). Learning monocular visual odometry with dense 3D mapping from dense 3D flow. *arXiv:1803.02286 [cs]*. arXiv: 1803.02286.