Robust Vision-Aided Self-Localization of Mobile Robots

Kyle Lindgren

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2020

Reading Committee:

Blake Hannaford, Chair

Jenq-Neng Hwang

E. Jared Shamwell

Program Authorized to Offer Degree:
Electrical & Computer Engineering

University of Washington

Abstract

Robust Vision-Aided Self-Localization of Mobile Robots

Kyle Lindgren

Chair of the Supervisory Committee:
Professor Blake Hannaford
Electrical & Computer Engineering

Machine learning has emerged as a powerful tool for solving many computer vision tasks by extracting and correlating meaningful features from high dimensional inputs in ways that can exceed the best human-derived modeling efforts. However, the area of vision-aided localization remains diverse with many traditional approaches (i.e. filtering- or nonlinear least-squares- based) often outperforming deep approaches and none declaring an end to the problem. Proven successes in both approaches, with model-free methods excelling at complex data association and model-based methods benefiting from known sensor and scene geometric dynamics, elicits the question: can a hybrid approach effectively combine the strengths of model-free and model-based methods? This work presents a new vision-aided localization solution with a monocular visual-inertial architecture which combines model-based optimization and model-free robustness techniques to produce scaled depth and egomotion estimates. Additionally, a Mixture of Experts ensemble framework is presented for robust multi-domain self-localization in unseen environments. Advancements in virtual environments and synthetically generated data with realistic imagery and physics engines are leveraged to aid exploration and evaluation of self-localization solutions. Access to diverse, abundant, and manipulable data also promotes the efficacy of transitioning simulator-tested solutions onto real-world vehicles, a rare ability for current deep approaches but a critical step for the advancement of the field. Together, well-established model-based techniques are combined with innovative model-free techniques to create a robust, hybrid, multi-domain self-localization

solution. Robustness to sensor, motion, and scene dynamics are demonstrated with comparison to state-of-the-art model-free and model-based approaches in both real and virtual domains.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

BA bundle adjustment 43

BooM Bootstrapped Monocular VIO 26

CNN convolutional neural network iii

EKF extended kalman filter 2

ICP iterative closest point 4

IMU inertial measurement unit iii

MAV micro aerial vehicle 26

MDN mixture density network 3

MoE Mixture of Experts iv

MSCKF multi-state constraint Kalman filter 2

OEC online error correction 18

RANSAC random sample consensus 5

SLAM simultaneous localization and mapping 1

SoA state-of-the-art 2

VIO visual-inertial odometry iv

VIOLearner Visual-Inertial-Odometry Learner 8

VISLAM visual-inertial simultaneous localization and mapping 28

VO visual odometry iv

VSLAM visual simultaneous localization and mapping 2

WTA winner-take-all 12

ACKNOWLEDGMENTS

I would like to express deep appreciation for my advisor Blake Hannaford, whose guidance and immense support through my early years of graduate school provided invaluable perspective and motivation. His encouragement to explore also enabled my transition to a full-time role with the Army Research Laboratory to broaden my studies. I would also like to thank my ARL mentor E. Jared Shamwell and his vast technical knowledge for always directing me in the right direction with a readily available chat.

Additionally, I would like to acknowledge current and former members of the UW BioRobotics Laboratory, Momona Yamagami, Kevin Huang, David Caballero, Astrini Sie, Bora Banjanin, and Yun-Hsuan Su, and members of the Army Research Laboratory, Sarah Leung and Chris Maxey for both their work as collaborators and providing an enjoyable grad school experience.

Chapter 1

INTRODUCTION

The term "visual odometry" refers to the methods by which motion observed through imagery provides localization estimates for dead-reckoning-based navigation [2]. The Mars Rover project [3, 4] employed visual odometry (VO) solutions in the 1980s, though the term did not become prominent in the research community until around 2004 [5]. Since then, the need for mobile robots to perform self-localization, as well as mapping (simultaneous localization and mapping (SLAM)), has grown dramatically. Safe and reliable navigation of autonomous systems is crucial to enable many impactful applications of robotics, including surveillance, supply chain, and emergency response to name a few. Extensive efforts over the past 30+ years have resulted in traditional methods which estimate egomotion with well-defined models of system dynamics, and while these mature methods perform well under many scenarios, they are limited in real-world applicability due to a lack of robustness to sensor and scene variations [6]. Separately, many recent learning-based techniques have emerged and instead rely on large amounts of exemplary data to drive the formation of a model-free solution to self-localization. While demonstrating previously unseen abilities, such as single-view depth estimation, the presence of several challenges limits learning-based methods from becoming the end solution to this topic. Lack of sufficient training data, a problem for learning-based methods, however, can critically benefit from emerging advancements in virtual environments capable of creating diverse, manipulable, and accurate imitations of the real-world. Consideration of virtual environment capabilities along with the respective strengths and weaknesses of model-based and model-free self-localization methods suggests that the most comprehensive real-world solution should incorporate both methodologies in a hybrid approach (See Appendix A for an exemplary scenario).

1.1 Background

1.1.1 Traditional Methods

In VO and visual simultaneous localization and mapping (VSLAM) in unknown environments, only data from camera sensors is used and tracked across frames to determine the change in the camera pose. SLAM approaches typically consist of a front end, in which features are detected in the image, and a back end, in which features are tracked across frames and matched to keyframes to estimate camera pose, with some approaches performing loop closure as well. ORB-SLAM2 [7] is a VSLAM system for monocular, stereo, and RGB-D cameras. It uses bundle adjustment and a sparse map for accurate, real-time performance on CPUs. ORB-SLAM2 performs loop closure to correct for the accumulated error in its pose estimation. ORB-SLAM2 has shown state-of-the-art (SoA) performance on a variety of VO benchmarks.

In visual-inertial odometry (VIO) and visual-inertial SLAM, the fusion of imagery and IMU measurements are typically accomplished by filter-based approaches or non-linear optimization approaches. The multi-state constraint Kalman filter (MSCKF) [8] has become a standard for filtering-based approaches to VIO. While its complexity is linear in the number of features being used for egomotion estimation and it is generally more robust than optimization-based approaches, MSCKF approaches are typically less accurate in comparison. ROVIO [9] is another filtering-based VIO algorithm for monocular cameras, which used a robust and efficient robocentric approach in which 3D landmark positions were estimated relative to the current camera pose. It used an extended kalman filter (EKF) to fuse the sensor data, utilizing the intensity errors in the update step. However, because ROVIO is a monocular approach, accurate scale is not recovered. OKVIS [10] is an optimization-based keyframe visual-inertial SLAM approach for monocular and stereo cameras. OKVIS relied on keyframes, which consisted of an image and estimated camera pose, a batch non-linear optimization on saved keyframes, and a local map of landmarks to estimate camera egomotion. VINS-Mono [11], another nonlinear optimization-based estimator, improves upon OKVIS with use of a loosely-coupled sensor fusion initialization procedure and relocalization from a tightly-coupled process as well as direct IMU integration, pose graph optimization, and

loop closure.

There are also several approaches which enhance VIO with depth sensing or laser scanning. Two methods of depth-enhanced VIO are built upon the MSCKF [8, 12] algorithm mentioned above. One method is the MSCKF-3D [13] algorithm, which used a monocular camera with a depth sensor, or RGB-D camera system. The algorithm performed online time offset correction between camera and IMU, which is critical for its estimation process, and used a Gaussian mixture model for depth uncertainty. Pang et al. [14] also demonstrated a depth-enhanced VIO approach based on MSCKF, with 3D landmark positions augmented with sparse depth information kept in a registered depth map. Both approaches showed improved accuracy over VIO-only approaches. Finally, Zhang and Singh [15] proposed a method for leveraging data from a 3D laser. The approach utilized a multi-layer pipeline to solve for coarse to fine motion by using VIO as a subsystem and matching scans to a local map. It demonstrated high position accuracy and was robust to individual sensor failures.

1.1.2 Learning-Based Methods

Recently, there have been several successful unsupervised approaches to depth estimation that are trained using reconstructive loss from image warping. Garg et al., Godard et al., and Zhan et al. [16–18] used stereo image pairs with known baselines and reconstructive loss for training. Thus, while unsupervised, the known baseline effectively provides a scale reference with a known transform between two images.

Pillai and Leonard [19] demonstrated visual egomotion learning by mapping optical flow vectors to egomotion estimates via a mixture density network (MDN). Their approach not only required optical flow to already be externally generated (which can be very computationally expensive), but also was trained in a supervised manner and thus required the ground truth pose differences for each exemplar in the training set.

SFMLearner [20] demonstrated the unsupervised learning of unscaled egomotion and depth from RGB imagery. They input a consecutive sequence of images and output a change in pose between the middle image of the sequence and every other image in the sequence, and the estimated

depth of the middle image. However, their approach was unable to recover the scale for the depth estimates or, most crucially, the scale of the changes in pose. Thus, their network's trajectory estimates needed to be scaled by parameters estimated from the ground truth trajectories and in the real-world, this information will of course not be available. SFMLearner also required a sequence of images to compute a trajectory. The recent work of [21] extended SFMLearner with a semi-differentiable iterative closest point (ICP) module.

UnDeepVO [22] is another unsupervised approach to depth and egomotion estimation. It differs from [20] in that it was able to generate properly scaled trajectory estimates. However, unlike [20] and similar to [16, 17], it used stereo image pairs for training where the baseline between images is known and thus, UnDeepVO can only be trained on datasets where stereo image pairs are available. Additionally, the network architecture of UnDeepVO cannot be extended to include motion estimates derived from inertial measurements because the spatial transformation between paired images from stereo cameras are unobservable by an IMU (stereo images are recorded simultaneously).

VINet [23] was the first end-to-end trainable visual-inertial deep network. While VINet showed robustness to temporal and spatial misalignments of an IMU and camera, it still required extrinsic calibration parameters between camera and IMU. In addition, VINet was trained in a supervised manner and thus required the ground truth pose differences for each exemplar in the training set which are not always readily available.

The recent work of [24] directly addressed the fusion of visual and inertial data in a deep VIO approach. They uncover domain specific robustness improvements to image and inertial noise as well as temporal misalignment with several fusion modalities while generating scaled estimates with monocular imagery. However, as in [23], their network was trained in a supervised manner, limiting its applicability.

While these and other deep approaches applied to vision-aided localization exhibit impressive learning capabilities, it is not trivial to incorporate effective structure into solutions, nor is it necessary to rely solely on entirely learned solutions.

1.1.3 Hybrid Approaches to Self-Localization

Several previous works have taken estimated scene depth images and recovered scale using geometry constraints between the camera and scene for frame-to-frame motion estimates. Choi et al. [25] employed a 1-point algorithm [26] which reduces the 3D (6 DoF) problem to 2D planar and circular motion (2 DoF) for ground vehicles. This novel approach resulted in significant error, though an online self-learning scheme proposed in [27] learns a ground classifier using a Bayesian framework which showed improved accuracy and reduced computation compared to standard random sample consensus (RANSAC) for monocular VO. Lastly, Song [28] achieved improved robustness to environmental information by using a more encompassing object tracking framework which considered triangulation of features, stereo matching, and object bounding boxes.

These works are most notable, though, for demonstrating the feasibility of recovering scale from unscaled depth, which neural networks can estimate from only monocular inputs. Yin et al. [29] circumvented the issues that arise from relying on the ground plane and camera height for scale recovery by instead considering the whole scene. Matched ORB features across two consecutive images in combination with depth image estimates from a neural network are iteratively refined to produce scaled monocular VO and depth estimates.

DPC-Net [30] addresses the problem of stereo visual odometry with an approach that fully melds traditional and learning-based tools. They incorporate convolutional neural network layers and supervised training to improve odometry estimation by focusing on *corrections* made by traditional VO methods.

1.1.4 Crossing Domains

Domain adaptation (DA), a subclass of transfer learning (TL), is concerned with the problem of making data from one or more source domains useful for completing new tasks in a target domain. For the case of large discrepancies between source and target domains, multi-step DA is often needed and addresses the issue by creating an intermediate domain to help bridge the gap [31]. The separately trained intermediate domain solution aims to identify and correlate features present

in both source and target domains.

Acquiring large amounts of diverse and accurately labeled training data for learning-based algorithms can be excessively onerous and result in data that excludes behaviors of interest. Modern computer graphics, however, are closing the gap between virtual and real-world generated imagery and sensor measurements, leading to creation of many publicly available synthetic datasets aimed at assisting deep learning.

Virtual KITTI [32] proposed a real-to-virtual world pipeline for generating driving data with accurate labeling and ground truths for object detection, tracking, scene and instance segmentation, depth, and optical flow. Real-world sample data provides initial calibration and construction of the virtual counterpart at which point the Unity game engine is used to build the full dataset with varying weather conditions.

For all applications, transferring behavior learned with simulated data to the real-world unsurprisingly presents many challenges, given the monumental task of capturing the infinitely complex analog world. Some works, however, have shown promising results with strategies that utilize both real and virtual training data.

Tsai et al. [33] explored the effects of using synthetically generated samples to train a neural network hand classifier. They concluded that adding a relatively small amount of real-world samples to their training dataset drastically improved recognition performance, and attributed this to the fact that the color of their synthetic samples greatly deviated from those of the real-world.

Additionally, Xie et al. [34] applied deep reinforcement learning with exclusively synthetic data to the task of obstacle avoidance and demonstrated successful transfer to the real-world. To shorten the domain gap and reduce illumination and appearance differences, virtual depth images are blurred and noised to more closely resemble real-world depth images captured with a Kinect.

The popular deep domain transferring work of [35] successfully demonstrates a domain-invariant neural network architecture for visual tracking through the use of several shared layers followed by domain-specific fully connected layers trained separately. For testing, the domain-specific layers are discarded and replaced by untrained layers that are trained online. While novel, the authors note effectiveness only for a simple task. The complexity of estimating ego-motion across diverse

domains presents challenges for a shared layer architecture.

The work of [36] uses synthetic KITTI data [37] with groundtruth optical flow and pose information to learn along with a style-transferring generative adversarial network (GAN) to learn and transfer a depth estimation net to the real KITTI dataset. The authors show substantial performance gain from leveraging synthetic data, but their domain transfer methods are limited by only operating from a groundtruthed synthetic environment to one visually similar real-world domain.

AD-VO promotes domain-robust VO estimation by uniquely excluding RGB imagery from its input and instead relying solely on depth images [38]. While promising, the work does not evaluate robustness across domains and additionally relies on the depth estimating framework from [17] which has the added constraint of requiring stereo imagery for its training data to learn crisp depth images that include fine structures.

1.1.5 Ensemble Methods

Ensemble methods have proven effective in a wide array of topic areas by utilizing multiple learners and a combination of their output to solve a single problem, as opposed to the typical strategy of refining one learner. Early ensemble research showed classification accuracy can be greatly improved by considering a set of classifiers [39], a set of 'weak learners' can result in a 'strong learner' through the use of now popular techniques such as AdaBoost [40] and Bagging [41], and a Mixture of Experts (MoE), divide-and-conquer strategy often demonstrates improved performance with neural network learners [42]. Several popular neural network tasks have made use of the MoE framework, including object detection, recognition, and tracking, but pose estimation remains a much less explored topic.

The VO work of [43] employs a mixture of experts framework for robust optical flow estimation but classifies expert domains by scene depth profiles within one dataset and does not include additional dataset evaluation. Their supervised approach also relies on groundtruth pose data for training.

AdapNet makes use of a modified MoE protocol, called Convoluted Mixture of Deep Experts (CMoDE), for the vehicle navigation subtask of semantic segmentation [44]. Impressive results

are shown across three datasets with varying weather conditions, but their network architecture includes a learned gating layer followed by shared layers, eliminating the ability to extend the solution to include additional experts.

1.1.6 VIOLearner Learning-Based Visual-Inertial Odometry Network

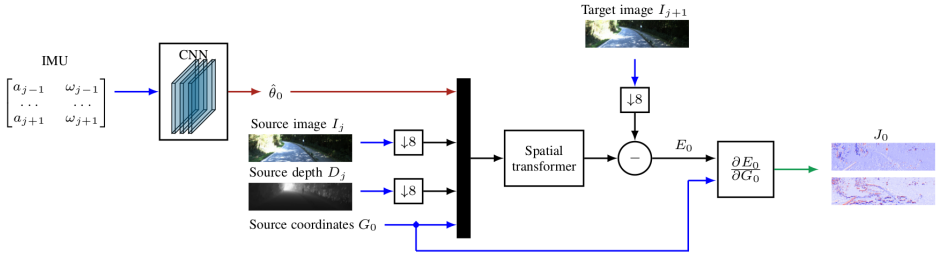
Visual-Inertial-Odometry Learner (VIOLearner), the self-localization neural network that served as the initial base framework built upon in this work was developed within the Electronics for Sense and Control group at the Army Research Lab and presented at the International Conference on Intelligent Robots and Systems (IROS) 2018 [45]. The following section details its approach and implementation, with the key features including:

- Built-in online error correction modules,
- Unknown IMU-camera extrinsics; and
- Loosely temporally synchronized camera and IMU

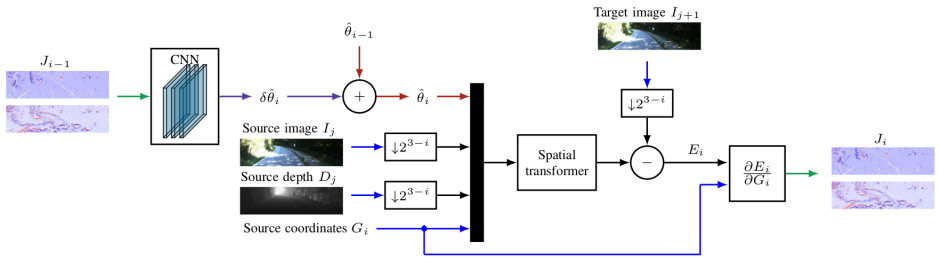
VIOLearner (Fig. 1.1) is an unsupervised VIO deep network that estimates the scaled ego-motion of a moving camera between some time t_j at which source image I_j is captured and time t_{j+1} when target image I_{j+1} is captured. VIOLearner receives an input RGB-D source image, a target RGB image, IMU data from t_{j-1} to t_{j+1}, and a camera calibration matrix K with the camera's intrinsic parameters. With access to K, VIOLearner can generate camera pose changes in the camera frame using a view-synthesis approach where the basis for its training is the Euclidean loss between a target image and a reconstructed target image generated using pixels in a source image sampled at locations determined by a learned 3D affine transformation (via a spatial transformer of [46]).

Multi-Scale Projections and Online Error Correction

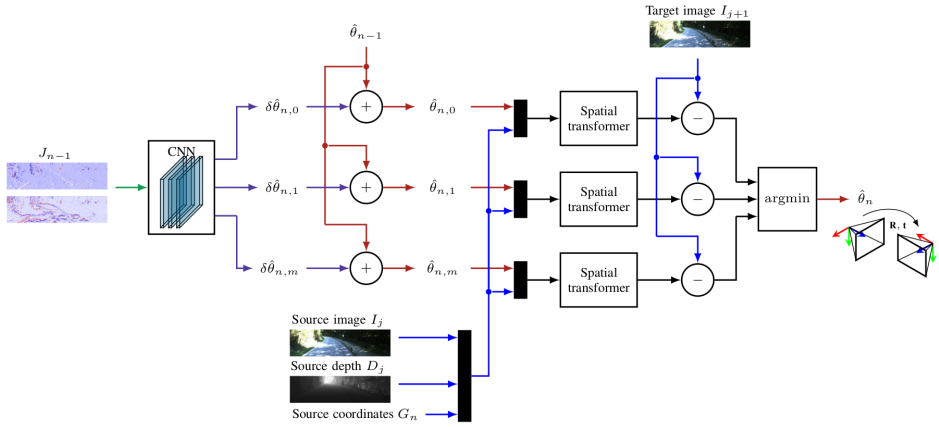
Similar to [20], VIOLearner performs multi-scale projections, with projections scaled by factors of 8, 4, and 2. However, with VIOLearner, multi-scale projections not only help to overcome gradient



(a) Level 0



(b) Level i



(c) Level n

Figure 1.1: VIOLearner: (a) IMU data is fed into a series of convolutional neural network (CNN) layers which output a 3D affine matrix estimate of change in camera pose $\hat{\theta}_0$ between a source image I_j and target image I_{j+1}. The transform $\hat{\theta}_0$ is applied to a downsampled source image via a spatial transformer module and the Euclidean error between the spatial transformation and the downsampled target image is computed as E_0. The Jacobian $\frac{\partial E_0}{\partial G_0}$ of the error image E_0 is taken with respect to the source coordinates G_0, and fed to the next level. (b) The Jacobian $\frac{\partial E_{i-1}}{\partial G_{i-1}}$ from the previous level is fed through CNN layers to produce an additive refinement $\partial\hat{\theta}_i$ that is summed with the previous transform estimate $\hat{\theta}_{i-1}$ to produce a new transform estimate $\hat{\theta}_i$. The Jacobian $\frac{\partial E_i}{\partial G_i}$ is propagated forward. (c) In the final level, the previous Jacobian $\frac{\partial E_{n-1}}{\partial G_{n-1}}$ is processed through CNN layers for m hypothesis refinements.

locality during training (See [20,47] for a broader discussion of overcomimg gradient locality), but also aid in online error correction at runtime.

Generally, at each level, the network computes a Jacobian of the reprojection error at that level with respect to the grid of coordinates. Convolutions are then performed on this Jacobian (sized HxWx2) and a $\delta\hat{\theta}$ is computed. This $\delta\hat{\theta}$ is added to the previously generated affine matrix $\hat{\theta}$. This is repeated for each of 5 levels in the hierarchy.

Generally, a Jacobian of a minimum size HxWx6 would be needed for traditional least-squares optimization. While the HxWx2 Jacobian only directly represents 2D information and does not include error with respect to depth, the convolutional processing of these Jacobians enables it to glean 3D information from this 2D representation and update the 6-DOF camera pose. In practice, we found that using this compressed HxWx2 2D Jacobian was an effective 6-DOF pose optimizer.

Level 0

VIOLearner first takes raw IMU measurements and learns to compute an estimated 3D affine transformation $\hat{\theta}_0$ that will transform a source image I_j into a target image I_{j+1} (the top left of Fig. 1.1a). The network downsamples ($\downarrow$ in Fig. 1.1) the source image I_j (and associated depth and adjusted camera matrix K for the current downsampling factor) by a factor of 8 and applies the 3D affine transformation to a normalized grid of source coordinates $[X_{src}, Y_{src}]$ to generate a transformed grid of target coordinates $[X_{tgt}, Y_{tgt}]$.

VIOLearner then performs bilinear sampling to produce a reconstruction image Ir by sampling from the source image I_j at coordinates $(x, y) \in [X_{tgt}, Y_{tgt}]$:

$$Ir(x,y)=\sum_{w,h}^{W,H} I_j(w,h)\max(0,1-|x-w|)\max(0,1-|y-h|) \tag{1.1}$$

As in [46], by only evaluating at the sampling pixels, we can therefore compute sub-gradients that allow error backpropagation to the affine parameters $\hat{\theta}_0$ by computing error with respect to the coordinate locations (x, y). For each pixel k, these Jacobians are computed as:

$$\frac{\partial Ir_k}{\partial x_k} = \sum_{w,h}^{W,H} I_j(w,h) \max(0, 1 - |y_k - h|) \begin{cases} 0, |w - x_k| \geq 1 \\ 1, w \geq x_k \\ -1, w < x_k \end{cases} \tag{1.2}$$

$$\frac{\partial Ir_k}{\partial y_k} = \sum_{w,h}^{W,H} I_j(w,h) \max(0, 1 - |x_k - w|) \begin{cases} 0, |h - y_k| \geq 1 \\ 1, h >= y_k. \\ -1, h < y_k. \end{cases} \tag{1.3}$$

Starting with Level 0, the Euclidean loss is taken between the downsampled reconstructed target image and the actual target image. For Level 0, this error is computed as:

$$E_0 = \|Ir^0 - I^0_{j+1}\|^2 \tag{1.4}$$

The final computation performed by Level 0 is of the Jacobian of the Euclidean loss of Equation 1.4 with respect to the source coordinates G_0 from Equation 1.2 and Equation 1.3. The resulting Jacobian matrix J_0 has the same dimensionality as the grid of source coordinates G_0 ($\frac{H}{8}$x$\frac{W}{8}$x2) and is depicted in Fig. 1.1a. In traditional approaches, the gradient and error equations above are only used during training. However, VIOLearner is novel in that it also computes and employs these gradients during each inference step of the network. During both training and inference, the Jacobian J_0 is computed and passed to the next level for processing.

Levels i to n-1

For the intermediate levels i through $n-1$, the previous level's Jacobian J_{i-1} is input and processed through layers of convolutions to generate a $\partial\hat{\theta}_i$. This $\partial\hat{\theta}_i$ represents a computed correction to be applied to the previous level's 3D affine transform $\hat{\theta}_{i-1}$. $\partial\hat{\theta}_i$ is summed with $\hat{\theta}_{i-1}$ to generate $\hat{\theta}_i$ which is then applied to generate a reconstruction that is downsampled by a factor 2^{3-i}. Error is again computed as above in Equation 1.4 and the Jacobian is similarly found as it was in Level 0

and input to the next Level $i + 1$.

Level n and Multi-Hypothesis Pathways

The final level of VIOLearner employs multi-hypothesis pathways similar to [48] where several possible hypotheses for the reconstructions of a target image (and the associated transformations $\hat{\theta}_m$, $m \in M$ which generated those reconstructions) are computed in parallel. The lowest error hypothesis reconstruction is chosen during each network run and the corresponding affine matrix $\hat{\theta}_{m*}$ which generated the winning reconstruction is output as the final network estimate of camera pose change between images I_j and I_{j+1}.

This multi-hypothesis approach allows the network to generate several different pathways and effectively sample from an unknown noise distribution. For example, as IMUs only measure linear accelerations, they fail to accurately convey motion during periods of constant velocity. Thus, a window of integrated IMU measurements are contaminated with noise related to the velocity at the beginning of the window. With a multi-hypothesis approach, the network has a mechanism to model uncertainty in the initial velocity.

Error for this last multi-hypothesis level is computed according to a winner-take-all (WTA) Euclidean loss rule (see [48] for more detail and justifications):

$$Ir^{n,*} \longleftarrow \underset{k}{\operatorname{argmin}} \|Ir^{n,k} - I^n_{j+1}\|^2 \tag{1.5}$$

$$E_n = \|Ir^{n,*} - I^n_{j+1}\|^2 \tag{1.6}$$

where $Ir^{n,*}$ is the lowest error hypothesis reconstruction. Loss is then only computed for this one hypothesis pathway and error is backpropagated only to parameters in that one pathway. Thus, only parameters that contributed to the winning hypothesis are updated and the remaining parameters are left untouched.

The final loss $\mathcal{L}$ by which the network is trained is then simply the sum of the Euclidean loss

terms for each level plus a weighted L1 penalty over the bias terms which we empirically found to better facilitate training and gradient backpropagation:

$$\mathcal{L} = \sum_{i=0}^{n} E_i + \lambda |bias| \tag{1.7}$$

Chapter 2
DEEP LEARNING WITH SYNTHETIC ENVIRONMENTS

Data-driven deep learning approaches are promising for uncovering abstract and complex relationships that manual and rule-based classification schemes fail to discover. Such an approach is amenable towards vision-aided self-localization, but requires myriad data which needs to be collected physically via extensive experiments. This process, however, is onerous and results in 'baked' data with limited ability to manipulate individual data properties through post-processing. A streamlined and controllable approach to dataset creation is needed. To that end, this chapter explores a novel method of synthetic dataset generation that leverages modern gaming engines. Results indicate that the development platform, despite being synthetic and requiring no physical data collection, facilitates critical component-level analysis of vision-aided self-localization methods.

2.1 Background

Many recent advances in computer vision areas leverage deep learning, as model-free approaches have proven more adept than model-based for solving highly complex tasks. One of the major challenges in this technique, however, is that it requires vast amounts of data for training. This training data must contain content rich enough to characterize anticipated variability in real-world implementations. Despite the existence of many large datasets readily available for vehicle navigation, these are relatively limited when comparing the level of variability contained. One way to alleviate this problem is by the process of data augmentation, which involves synthetically generating dense datasets from currently available sparse datasets, leading to more effective training and solution analysis [49].

To conduct a detailed analysis of learning-based visual odometry components, a virtual environment experimentation platform has major practical advantages, such as

- access to absolute ground truth data;
- control of individual factors;
- experiment repeatability; and
- efficient dataset creation.

Further, synthetic data collection with virtual environments can potentially enable learning of behaviors not captured with available real-world datasets [50], shortening the timeline for real-world implementation.

VIOLearner [45] employs an online error correction module and a multi-hypothesis mechanism (Fig. 1.1) in its deep neural network architecture that learns and corrects sensor misalignments during operation. Here, we extend the analysis of VIOLearner with a more challenging aerial vehicle dataset and exploration of the effects of IMU-camera extrinsic offsets on trajectory estimation. In particular, in [45], it was theorized that VIOLearner performed favorably on the KITTI Odometry dataset [51] due to its ability to generate multiple, distinct pose estimates for every input set while also learning how to correct errors in its multi-scale levels online. The analysis here aims to further test this hypothesis by evaluating the robustness of VIOLearner's error correction modules to changes in IMU-camera extrinsics.

2.2 Methods

2.2.1 Virtual Environment

An experimental environment (Fig. 2.1) was built with Unreal Engine, a game development platform which enables realistic, high-fidelity visualizations and physics simulations. Drone flight and control leverage the Microsoft AirSim plugin [52] for simulated flight dynamics as well as a suite of sensors including inertial (GPS, barometer, gyroscope, accelerometer) and imaging (RGB, depth, segmentation). For this work, $128\text{x}480$ resolution RGB-D imagery is captured at 10 Hz with IMU measurements taken at 100 Hz.

The environment utilized simulates a maze-like block arena, denoted *Blocks*. A total of 20 trajectories were captured within the *Blocks* environment, with each trajectory containing 2 minutes of flight time. The dataset was split with 70% of trajectories in the training set, 15% in the validation set, and 15% in the test set.

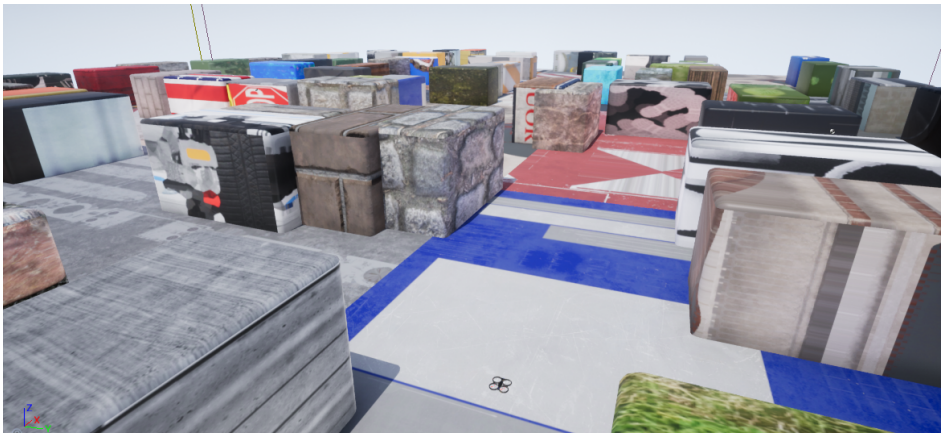


Figure 2.1: The virtual *Blocks* environment exhibits visually unique imagery through use of an array of object materials.

2.2.2 Trajectory Characteristics

Manual flight control through the environment provided turning and velocity inputs for each trajectory. AirSim managed drone dynamics such that turns and changes in velocity created realistic roll, pitch, and yaw motion. Velocity was kept between 2 m/s and 4 m/s while vertical positioning of the aerial vehicle remained fixed throughout all flights.

2.2.3 Generating IMU-Camera Configurations

Post-processing of AirSim-generated ground truth vehicle kinematics facilitates analysis of network performance on trajectories that differ in IMU configurations (position and orientation on the vehicle), while keeping all other state information exactly the same. Additionally, vehicle linear accelerations and angular velocities for any IMU position and orientation configuration can be

calculated without the need for flight playback.

Linear acceleration observed by any point on the rigid body vehicle can be calculated using the linear acceleration a, angular velocity ω, and angular acceleration α vectors of a known point on the vehicle, in addition to the transformation matrix θ, containing the rotation matrix R and translation vector T between the two points [53]. Linear acceleration of the desired point is calculated by $\hat{a} = a_T + a_R$, where the tangential component a_T equals the linear acceleration of the known point, and the radial component a_R is found from the derivative of linear velocity with respect to time.

$$v = \omega \times T$$

$$a_R = \frac{dv}{dt} = \frac{dw}{dt} \times T + w \times \frac{dT}{dt} \tag{2.1}$$

where

$$\frac{dw}{dt} = \alpha, \ \frac{dT}{dt} = v,$$

giving

$$a_R = \alpha \times T + w \times v \tag{2.2}$$

Angular velocity, the other IMU measurement of interest here, is shared among all points on a rigid body, so $\hat{\omega} = \omega$.

Noise is added to both generated measurements to increase similarity to real-world sensor performance. The noise model injects two types of sensor errors: white noise n, and sensor bias b. A noised linear acceleration measurement, for instance, is generated by

$$\tilde{a}(t) = a(t) + b(t) + n(t) \tag{2.3}$$

with both sensor error types sampled at each time t.

Lastly, generated linear accelerations and angular velocities undergo rotations according to R to account for changes in orientation between the known IMU and desired poses.

2.3 Evaluation

As was affirmed in the recent work of Delmerico et al. [54], optimization based approaches to VIO (e.g. OKVIS [10]) typically outperform stochastic filtering based approaches (e.g. MSCKF [8]). We have thus chosen OKVIS [10] as the primary comparator to SoA VIO approaches.

2.3.1 Online Error Correction

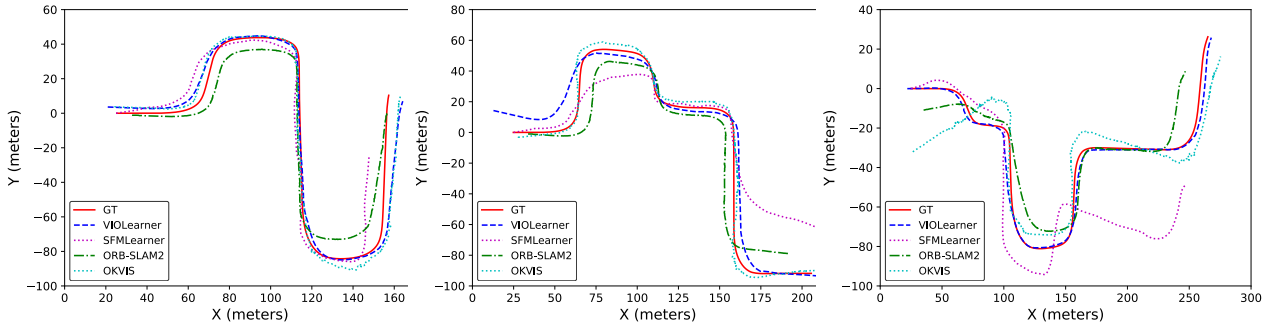
The online error correction (OEC) modules were designed to perform error correction at runtime, and thus able to correct mistakes stemming from either uncertain/degraded sensory information, or from novel (unseen) sensory information. But as is the case with any deep network, overfitting is always a concern: how do we know that the OEC modules are actually learning to perform a dynamic function and not simply overfitting to the domain? To evaluate the OEC modules, we systematically augmented the camera-IMU extrinsics (pose offset) of while keeping all other trajectory characteristics the same. We trained networks on data with a given camera-IMU extrinsic calibration and then manipulated the extrinsic calibration for testing (as described in Section. 2.2.3). By varying these extrinsics, we are able to generate camera-IMU misalignments which result in effects visible in the Level 0 prediction of $\hat{\theta}_0$ (the first estimate of pose change generated by the network that comes entirely from IMU data). Results on the real-world KITTI driving dataset are also included for reference (Tab. 2.1).

2.3.2 Evaluation Metrics

We evaluate our trajectories primarily using the standard KITTI relative error metric (reproduced below from [51]):

$$E_{rot}(F) = \frac{1}{F} \sum_{(i,j)\in F} \angle[(\hat{p}_j \ominus \hat{p}_i) \ominus (p_j \ominus p_i)] \tag{2.4}$$

$$E_{trans}(F) = \frac{1}{F} \sum_{(i,j)\in F} \|(\hat{p}_j \ominus \hat{p}_i) \ominus (p_j \ominus p_i)\|_2 \tag{2.5}$$



(a) AirSim Seq. 1 (b) AirSim Seq. 2 (c) AirSim Seq. 3

Figure 2.2: Trajectories for AirSim sequences 0, 1, and 2. Ground truth (GT) and results from VIOLearner, SFMLearner, ORB-SLAM2, and OKVIS are shown.

where F is a set of frames (i, j), $\hat{p} \in SE(3)$ and $p \in SE(3)$ are estimated and true camera poses, respectively, $\ominus$ is the inverse compositional operator, and $\angle[\bullet]$ is the rotation angle.

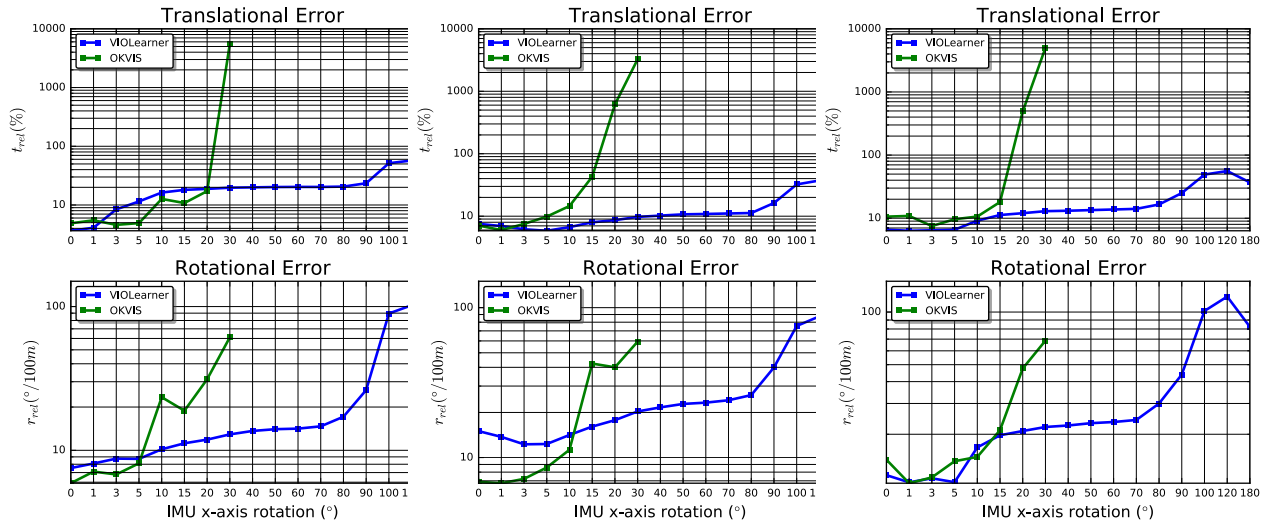
For KITTI, we use lengths of 100, 200, 300, 400, 500, 600, 700, and $800\,\mathrm{m}$. For AirSim, which has far shorter trajectories, we evaluate on lengths of 25, 50, 75, and $100\,\mathrm{m}$.

2.4 Results and Discussion

Table 2.1: Absolute trajectory estimation

Dataset	Traj	VIOLearner		SFMLearner	
		$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$
Blocks	1	3.64	7.52	21.97	17.61
	2	7.54	15.02	20.39	14.03
	3	6.60	11.67	17.60	19.06
Forest	1	2.84	7.40	80.20	158.96
	2	4.12	8.75	83.22	185.35
	3	3.13	7.69	79.04	162.57
KITTI*	09	1.51	0.90	21.63	3.57
	10	2.04	1.37	20.54	10.93

*KITTI error metrics were calculated on lengths of 100 m - 800 m, whereas Blocks and Forest were calculated on lengths of 25 m - 100 m.



(a) AirSim Seq. 1 (b) AirSim Seq. 2 (c) AirSim Seq. 3

Figure 2.3: Results on VIOLearner and OKVIS trajectory estimation on the *Blocks* environment given induced IMU translation and orientation offsets in the x-dimension. Measurement errors are shown for each sequence with translational error percentage (top row) and rotational error in degrees per $100\,\mathrm{m}$ (bottom row) computed across trajectory lengths $25\,\mathrm{m}$ - $100\,\mathrm{m}$. In contrast to VIOLearner, after $20-30^\circ$ (depending on the sequence), OKVIS exhibits catastropic failure in translation estimation.

2.4.1 Spatial Misalignments

The AirSim-simulated spatial misalignments between different camera and IMU configurations provide a method for evaluating the effectiveness of VIOLearner's online error correction (OEC) modules. An analysis was first performed with VIOLearner on both virtual environment datasets, followed by comparison with a SoA model-based VIO approach.

Orientation offsets (Fig. 2.3 bottom row) show a sharp increase in error around the 90 degree mark. This is plausibly expected since rotations of this magnitude result in complete dimension shift. The KITTI IMU coordinate system (x-forward, y-left, z-up) was used in this work, causing a positive 90 degree rotation about the x-dimension to shift y measurements to the input where the network was trained to expect the z measurements, with z becoming negative y. Unsurprisingly,

the network appears unable to compensate for dimension shifts.

Orientation shifts within a more reasonable range of less than 80 degrees, however, display a modestly sloped plateau that suggests successful online error correction. Further, realistic shifts of less than 10 degrees show lesser amounts of error and greater applicability to real-world implementation.

Position offsets (Fig. 2.3 right column) exhibit reduced effects on trajectory estimation, as compared to orientation offsets, even with 10 meter shifts. Therefore, VIOLearner is likely more robust to manipulation of IMU magnitudes (lever arm effect) than to IMU rotations.

Additionally, VIOLearner and SFMLearner show decreased performance on the virtual environment dataset compared to KITTI, particularly with respect to orientation. As was noted previously, however, KITTI was captured with a ground vehicle, whereas this work used an aerial vehicle. Despite limiting the aerial drone from changing altitude, changes in direction and velocity resulted in IMU measurements with greater sensor activity compared to ground vehicles. 'Opening' the vehicle pitch dimension effectively increased the input domain and complexity.

Comparison to OKVIS

Figure 2.2 displays qualitative results with several other self-localization methods on AirSim trajectories that do not include spatial misalignments. Figure 2.3 displays imu-camera orientation misalignment results for VIOLearner and OKVIS each of the *Blocks* test trajectories.

In contrast to VIOLearner's modestly sloped plateau with offsets less than 80 degrees, OKVIS showed an inability to handle orientation offsets greater than 30 degrees, despite surprising robustness to rotation errors under 20 degrees.

2.4.2 Robustness

To evaluate the ability of the VIOLearner architecture and its OEC modules to generalize, we performed additional experiments with regard to robustness to motion and image perturbations/domain shifts.

(a) No Fog

(b) Medium Fog

(c) Heavy Fog

Figure 2.4: Sample images from the three fog conditions.

Robustness to Motion

While the AirSim dataset exhibits greater motion variability than the KITTI driving dataset (in particular rotational variability), the training trajectories and evaluation trajectories presented thus far were predominantly forward moving with slow, car-like turns (see Fig. 2.2). To evaluate the ability of VIOLearner to generalize to previously unseen trajectories very different from those that the network was trained on, we collected two additional evaluation trajectories. In the first trajectory, referred to as *figure-eight*, the simulated quadcopter flew in a figure-eight pattern in AirSim in the same Blocks environment. In the second trajectory, referred to as *rotation-only*, the simulated

Table 2.2: Robustness evaluation of VIOLearner to shifts in the visual domain. Trajectory estimation from VIOLearner RGB-D, OKVIS, and ORB-SLAM2 on AirSim sequences 1, 2, and 3 with light fog ($foglevel = 0.05$), medium fog ($foglevel = 0.1$), and heavy fog ($foglevel = 0.3$). An X indicates that the algorithm failed to successfully complete the sequence.

	VIOLearner		OKVIS-S		ORB-SLAM2-S	
Seq						
	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$
1 (No Fog)	**3.19**	6.61	4.87	5.9	12.4	**1.93**
2 (No Fog)	**6.6**	13.45	7.08	6.89	13.25	**2.3**
3 (No Fog)	**5.04**	10.67	10.59	14.31	14.64	**5.5**
1 (Light Fog)	**6.45**	11.11	7.1	7.69	12.14	**2.03**
2 (Light Fog)	**11.07**	21.45	11.38	11.35	13.47	**1.44**
3 (Light Fog)	**9.38**	19.04	16.2	18.6	15.03	**6.59**
1 (Med. Fog)	8.86	16.42	**6.78**	**5.79**	X	X
2 (Med. Fog)	13.82	26.9	18.78	26.95	**13.43**	**2.8**
3 (Med. Fog)	**12.2**	23.65	14.17	15.24	13.02	**4.77**
1 (Heavy Fog)	**15.65**	28.82	15.7	**16.91**	X	X
2 (Heavy Fog)	**20.01**	**39.96**	21.16	47.34	X	X
3 (Heavy Fog)	**18.11**	33.65	33.95	**21.22**	X	X

quadcopter yawed in-place without translating. Results for these two trajectories are shown in Tab. 2.3. VIOLearner outperformed the stereo version of OKVIS and ORB-SLAM2 (OKVIS-S, ORB-SLAM2-S) on the *figure-eight* trajectory in translation error but was outperformed by OKVIS-S on the *rotation-only* trajectory.

Robustness to Imagery

VIOLearner implicitly learns the extrinsic parameters of the camera-IMU system as well as IMU intrinsics. As such, we have yet to evaluate a VIOLearner network trained in one domain (e.g.

Table 2.3: Robustness evaluation of VIOLearner to novel trajectories/motions not seen during training. Trajectory estimation from VIOLearner RGB-D, OKVIS, and ORB-SLAM2 on a figure-eight trajectory and a rotation-only trajectory in AirSim.

	VIOLearner (RGB-D)		OKVIS-S		ORB-SLAM2-S	
Seq						
	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$	$t_{rel}(\%)$	$r_{rel}(^\circ)$
Figure-Eight	**4.7**	11.88	5.2	5.14	13.38	**1.84**
Rotation-Only	9.08	21.57	**5.18**	4.51	13.5	**2.66**

KITTI) on a different domain (e.g. AirSim). To evaluate the ability of VIOLearner to generalize to previously unseen imagery and handle domain shifts, we performed additional experiments where we recollected the three AirSim evaluation trajectories from Fig. 2.2 using identical motion profiles but varying the weather-conditions (see Fig. 2.4). Results for these two trajectories are shown in Tab. 2.2. In the lightest fog condition, VIOLearner outperformed OKVIS-S and ORB-SLAM2-S in translation error. For the medium fog condition, VIOLearner was outperformed on two of the three trajectories. For the highest fog condition, ORB-SLAM2-S failed to run on any of the trajectories while VIOLearner outperformed OKVIS-S in translation error.

2.5 Conclusion

The methods used to generate training data for VIOLearner in this work presents a path towards efficient and improved state estimation of real-world robotic systems. Self-localization methods can benefit from the use of synthetic data by isolating and manipulating individual dataset components, bringing previously hidden associations to light. Additionally, motion and imagery robustness tests suggest VIOLearner is more capable to generalize solutions than the model-based methods it was compared against. Lastly, collecting training data within virtual environments using representative sensors for the desired testing system requires considerably less effort than real-world data collection. For the case that camera parameters such as resolution and FOV differ from previously

collected trajectories, previously captured control inputs can be used to efficiently recapture each flight trajectory with modified imaging. Differences only in IMU parameters, however, do not require additional flying time as was shown previously, provided the desired IMU pose on the vehicle and noise parameters are given.

Chapter 3

MODEL-BASED SCALE ESTIMATION WITH MODEL-FREE SELF-LOCALIZATION

Common deep learning approaches to vision-aided self-localization heavily rely on convolutional neural networks (CNNs) to extract relevant state information from raw sensors inputs, including inertial (GPS, barometer, gyroscope, accelerometer) and imaging (RGB, depth, segmentation) among others. While several deep approaches applied to this domain have demonstrated impressive learning capabilities, it is not necessary to rely solely on entirely learned solutions, particularly in instances where model-free approaches are employed where models are known. This chapter presents Bootstrapped Monocular VIO (BooM), a scaled monocular visual-inertial odometry (VIO) solution that leverages the complex data association ability of model-free approaches with the ability to exploit known geometric dynamics with model-based approaches. Our end-to-end, unsupervised deep neural network simultaneously learns to perform visual-inertial odometry and estimate scene depth while scale is enforced through a loss signal computed from position change magnitude estimates from traditional methods. We evaluate our network against a state-of-the-art (SoA) approach on the KITTI driving dataset as well as a micro aerial vehicle (MAV) dataset that we collected in the AirSim simulation environment. We further demonstrate the benefits of our combined approach through robustness tests on degraded trajectories.

3.1 Introduction

Traditional approaches to visual-inertial odometry (VIO) have exhibited remarkable performance in numerous applications, partially through the exploitation of closed-form geometric relationships between input imagery and IMU data. However, despite performing well overall, occasional failures can critically derail trajectory estimates due to insufficient image features, sensor misalign-

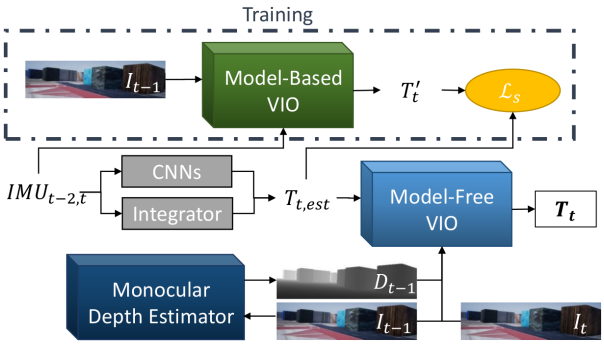


Figure 3.1: Overview of the BooM architecture. A model-free VIO network combines with a depth estimating network and translation magnitude estimates from a model-based VIO method to learn scaled estimates of egomotion. Through this approach, a model-based VIO method guides the model-free deep learning neural network towards the proper scale during training, which is retained and applied during inference.

ment and drift, and dynamic scenes often present in harsh environmental conditions. While deep approaches to VIO do not possess immunity to such scenarios, inherent robustness is observed from a lack of stringent constraints which often lead to catastrophic failures in classical schemes.

While we have previously demonstrated robustness to sensor misalignment with our deep VIO neural network [55], this work aims to leverage the imagery robustness strengths of deep approaches with those of traditional approaches. Specifically, we create a scaled monocular deep neural network VIO solution that is trained in an unsupervised manner with scale effectively bootstrapped during the training process with the help of a traditional VIO method (Fig. 3.1). Scale is enforced through a loss term derived from the magnitude of a traditional method's translation estimate. The constructed framework is shown to benefit from the robustness and complex data association abilities of deep neural networks while gaining the benefits of strict measurement models with classical approaches in two varying datasets.

In this paper, we present an end-to-end trainable monocular VIO approach which enforces scale from translation magnitude estimates computed from a traditional VIO approach. To the best of the authors' knowledge, this work is the first to

1. Integrate traditional VIO methods into the training process of an unsupervised deep monocular VIO method

2. Estimate scaled trajectories from a purely deep monocular VIO method

3.2 Background

A wide array of approaches with varying input modalities exist for the task of odometry estimation. In VO and VSLAM, camera sensors serve as the only input modality to determine the change in the camera pose. VSLAM approaches typically split the problem into feature tracking and keyframe tracking in which frames with high observances of strong image features aid camera pose estimation through matching across several frames. While VO and VSLAM can be performed by a monocular camera system, only scaled rotation can be recovered due to depth being unobservable. Monocular VO approaches, therefore, estimate translation with an arbitrary scale [56–59] and are severely limited in real-world applicability.

In VIO and visual-inertial simultaneous localization and mapping (VISLAM), scaled estimates are critically gained through the fusion of imagery and IMU measurements typically by filter-based approaches or non-linear optimization approaches. As IMUs measure only linear accelerations and angular velocities, inertial approaches to localization are prone to exponential drift over time due to the double integration of accelerations into pose estimates. Incorporating visual estimates of pose change with those of inertial methods, however, leads to reduced drift. For VIO, integrating raw IMU measurements can only provide noisy, short-term estimates of scale.

While traditional VIO approaches reliably perform well in ideal conditions, failures often occur due to featureless areas, motion blur, large viewpoint changes, dynamic elements in the environment and illumination changes. A primary cause of such shortcomings can be attributed to the use of hand-engineered features and heuristics. Additionally, limitations arise from stringent IMU intrinsics and IMU-camera extrinsics requirements. As such recently developed machine learning techniques have become increasingly attractive to VIO for their abilities to learn powerful data representations with increased robustness.

DPC-Net [30], introduced in Section 1.1.3, is a stereo visual odometry approach that also melds traditional and learning-based tools. This work is differentiated by utilizing inertial sensor data, monocular imagery, and unsupervised training. Further, our methods are closer to a deep approach

assisted by traditional modeling, whereas they assist traditional modeling with deep learning.

3.2.1 VIOLearner

The core architecture of this work builds upon VIOLearner, our unsupervised VIO deep network that estimates the scaled egomotion of a moving camera between two time instances. VIOLearner utilizes a view-synthesis approach and incorporates a multi-scale projection architecture similar to [20] to reduce gradient locality issues during training and also improve online error correction at runtime. Please see [45] for an in-depth description of VIOLearner.

3.3 Approach

BooM is an unsupervised monocular VIO deep network that estimates the scaled egomotion of a moving camera between some time t_{j-1} and time t_j, the two most recent images captured by the camera. BooM receives two RGB images (I_{j-1}, I_j), accelerometer and gyroscope data from the IMU ($a_{j-2,j}$, $w_{j-2,j}$), and the camera's intrinsic parameters (K matrix). With access to K, BooM can generate camera pose changes in the camera frame using a view-synthesis approach where the basis for its training is in the Euclidean loss between a target image and a reconstructed target image generated using pixels in a source image sampled at locations determined by a learned 3D affine transformation (via a spatial transformer of [46]). The 3D affine transformation is facilitated by an integrated depth network based upon SFMLearner [20] which learns to produce a depth image from the input monocular imagery (Fig. 3.3). Additionally, the IMU intrinsics and IMU-camera extrinsics are utilized with a traditional VIO approach during training to enforce scale on the simultaneously learned depth images.

3.3.1 Initial Pose Change Estimate

As is indicated by two gray modules in Fig. 3.1, accelerometer measurements from the preceding two time intervals of IMU data ($a_{j-2,j}$, $w_{j-2,j}$) undergoes a series of convolutions to generate an initial change in position estimate while direct integration of the gyroscope measurements reliably

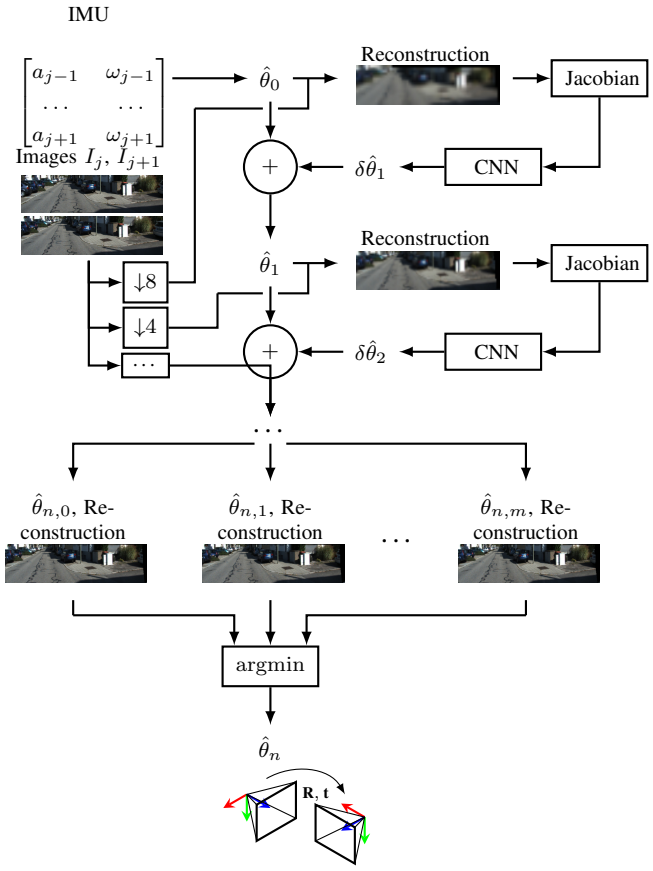


Figure 3.2: Generalized overview of our VIOLearner network. Note the hierarchical architecture where θ, the true change in pose between images, is estimated at multiple scales and then corrected via convolutional processing of the Jacobian of Euclidean projection error at that scale with respect to a grid of source coordinates. The final Level n computes m hypothesis reconstructions (and associated estimates $\hat{\theta}_{n,m}$) and the $\hat{\theta}_{n,m}$ with the lowest paired Euclidean projection error between the reconstruction and true target image I_{j+1} is output as the final network estimate $\hat{\theta}_n$.

Figure 3.3: Sample images from the AirSim MAV dataset showing the RGB 480x128 resolution images in the left column and the learned depth image counterparts in the right column.

yields accurate estimates of change in orientation. These two estimates combine to create the initial change in pose estimate $T_{t,est}$.

3.3.2 Bootstrapping Scale

Traditional VIO methods are used to provide vehicle translation estimates at time t_j using the image and IMU data at time t_j along with the camera's intrinsic parameters, IMU intrinsics, and IMU-camera extrinsics. These translation estimates are only needed during training, and are employed as a loss signal to steer BooM's depth subnetwork towards the correct scale. The progression of this process is shown in Fig. 3.4 with results gradually approaching and remaining fixed to the ground truth. The scale loss $\mathcal{L}_s$ is simply the difference in magnitude of translation estimates at time t_j between the model-free VIO estimate $p_{j,mf}$ and the model-based VIO estimate $p_{j,mb}$. The scale loss terms are summed for each level n in the core VIO framework.

$$\mathcal{L}_s = \sum_{i=0}^{n} ||p_{j,mb}| - |p_{j,mf}|| \tag{3.1}$$

For this work, we have elected to use VINS-Mono [11] to provide scale estimates for its proven success and wide adoption in the research community. Trajectory translation estimates were therefore collected by running VINS-Mono on all training sequences and saving the computed odometry results.

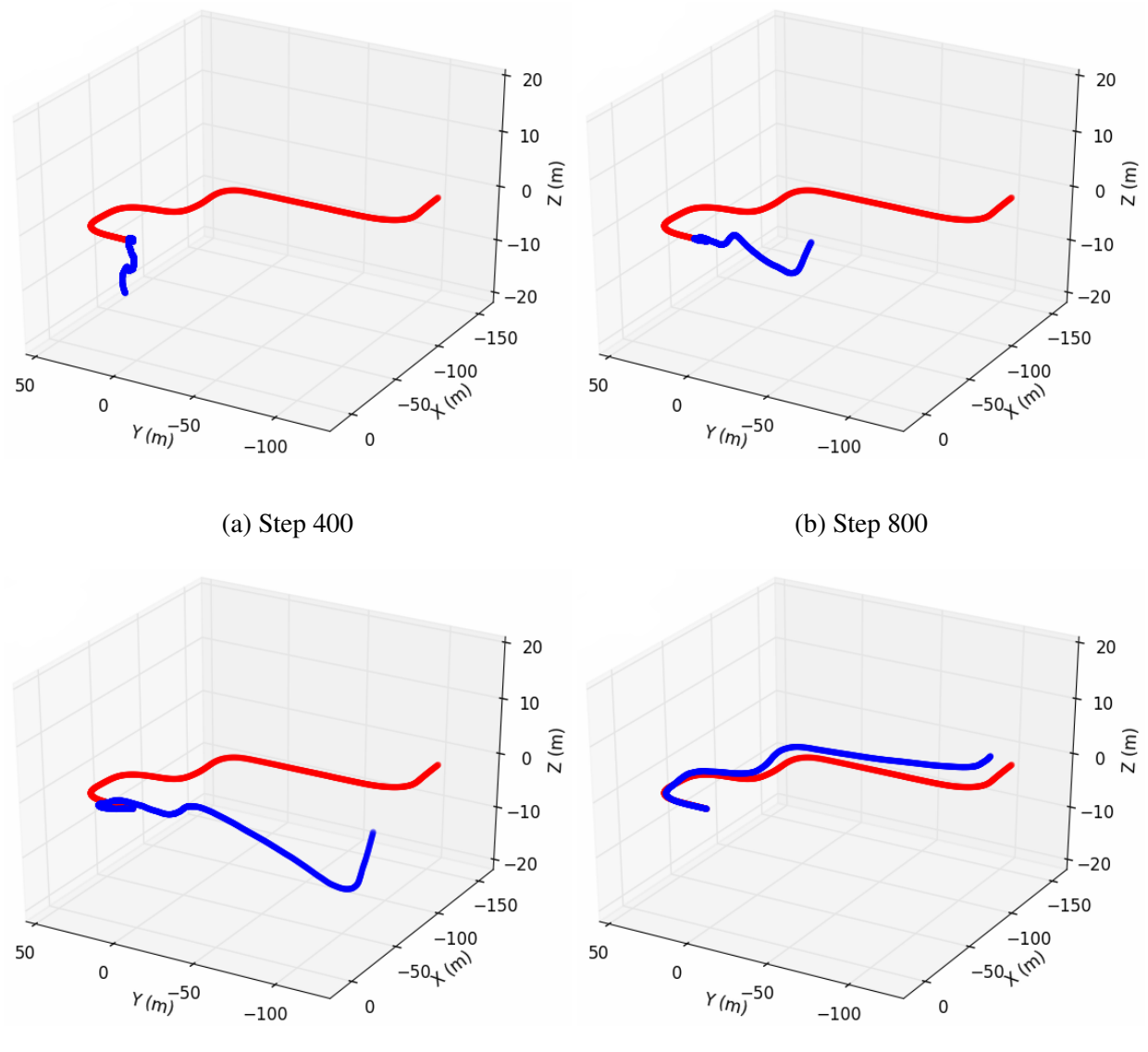
3.4 Methods

3.4.1 Training Procedures

All of the trained networks underwent similar training procedures. Each was trained using a batch size of 16 and allowed to train for at most $200,000$ iterations with an evaluation occurring every $1,000$ iterations on the respective validation set. We used the Adam solver with momentum1=0.9, momentum2=0.99, gamma=0.5, learning rate=$2\mathrm{e}{-5}$, and an exponential learning rate policy. The networks were trained using single-point precision (FP32) on desktop computers with 2.60 GHz Intel i9-7980XE processors and Nvidia GPUs.

3.4.2 KITTI Driving Dataset

We have elected to use the popular KITTI odometry dataset [51] to evaluate our approach on a ground vehicle platform. The KITTI dataset includes RGB and depth data captured over a driving distance of $39.2\,\mathrm{km}$ around a residential area. All 11 trajectories were ground truthed with accuracies within $10\,\mathrm{cm}$ via an OXTS RT 3003 GPS solution. Even though depth is provided, we do not utilize this input as the goal of BooM is to recover scale and be applicable to platforms with minimal hardware, specifically, only monocular RGB/grayscale cameras and IMU sensors.



(a) Step 400

(b) Step 800

(c) Step 1200

(d) Step 8400

Figure 3.4: Progression of scale learned on a sample AirSim validation trajectory.

Figure 3.5: The virtual city environment exhibits realistic lighting and texture of a nondescript city.

3.4.3 AirSim MAV Simulation Environment

An experimental environment (Fig. 3.5) built within Unreal Engine, a game development platform which enables realistic, high-fidelity visualizations and physics simulations, was used to support creation of a MAV dataset. Aerial vehicle control leverages the Microsoft AirSim plugin [52] for simulated flight dynamics as well as a suite of sensors including inertial (GPS, barometer, gyroscope, accelerometer) and imaging (RGB, depth, segmentation). Absolute groundtruth is provided by the virtual environment, facilitating near perfect error measures. For this work, 128x480 resolution RGB imagery is captured at 11.11 Hz with IMU measurements taken at 111.11 Hz.

Manual flight control through the environment, which resembles a city block, provided turn and velocity inputs for each trajectory, with AirSim managing drone dynamics such that turns and changes in velocity apply realistic effects to roll, pitch, and yaw motion. Velocity was kept between 0 m/s and 8 m/s during all flights. A total of 14 trajectories were captured within the environment, with each trajectory containing 108 seconds of flight time. The dataset was split with 10 of the trajectories in the training set, 2 in the validation set, and 2 in the test set.

3.4.4 Simulating Harsh Environments

The test trajectories of both the KITTI and AirSim datasets underwent augmentations to resemble real-world atmospheric conditions which often prove problematic for self-localization methods (Fig. 3.6). The Adobe After Effects software facilitated the addition of rain, fog, and snow sim-

ulations with varying intensities for robustness testing. All three atmospheric conditions included realistic movement and variation, though complementary environment factors such as the inclusion of clouds and a darkening of the scene for the rain and snow conditions was not incorporated. The size of the rain and snow particles were chosen as the parameter varied for differing intensities with opacity serving as the fog intensity variable. The intensity distributions for the fog and snow conditions were kept the same when applied to both the KITTI and AirSim datasets with the rain condition being intensified greater for AirSim. All levels were applied such that the performance of BooM could be evaluated against conditions where VINS-Mono began exhibiting significant difficulties.

Figure 3.6: Sample images from the test trajectories with simulated harsh conditions. KITTI images are on the left with AirSim on the right. The top row is the unaltered image followed by the rain, fog, and snow conditions.

3.5 Evaluation

We evaluate BooM on both the KITTI and AirSim datasets with VINS-Mono [11] providing VIO comparison and representing the SoA traditional method. Performance comparison on the test trajectories assesses the feasibility of the proposed approach, particularly if scale can be steered during training and applied during inference.

The KITTI and AirSim test sequences were separately degraded with the addition of rain, fog, and snow (Fig. 3.6) to directly test the believed robustness advantage of deep VIO approaches over traditional approaches, and indicate BooM's successful combination of model-based benefits with those of mode-free methods.

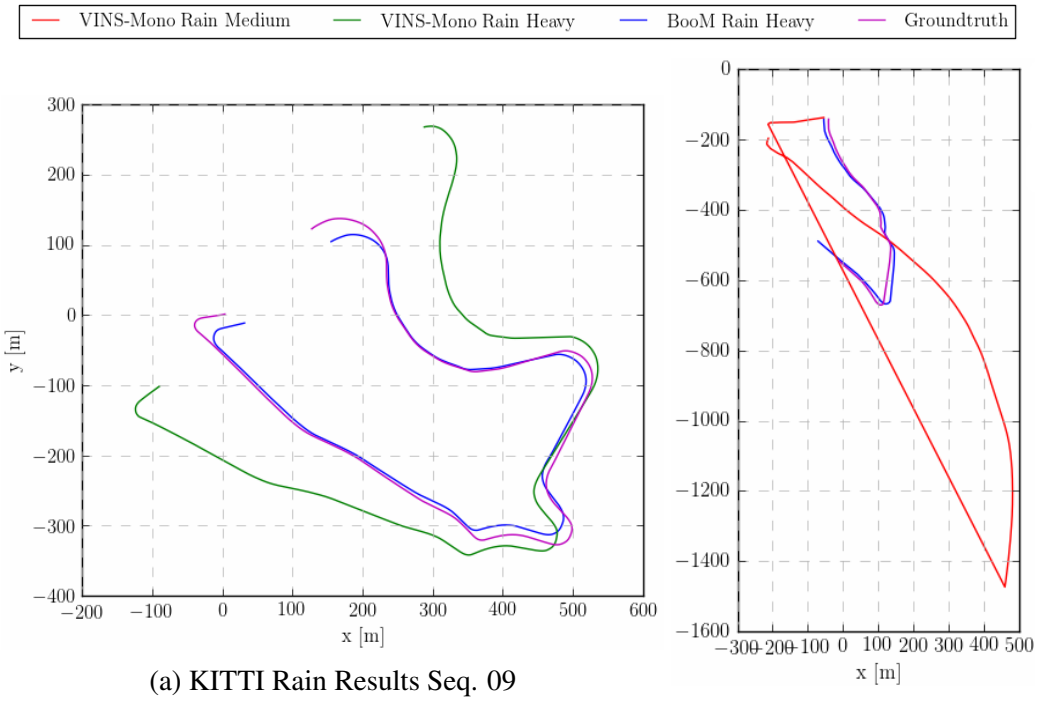
To allow for fair comparison, approximately 20 seconds at the beginning of each trajectory is discarded from evaluation to allow a reasonable amount of time for VINS-Mono to initialize. Additionally, with respect to the KITTI dataset, the full resolution 1241x376 imagery was provided to VINS-Mono for all sequences with BooM utilizing scaled and cropped 480x128 resolution images.

Error metrics are computed using the RPG trajectory evaluation toolkit [60] for absolute translation in RMSE values and relative rotational yaw error. Relative rotational error is chosen over absolute rotational error to provide a more useful metric for VIO approaches which do not contain a global reference. All trajectories are aligned in the customary visual-inertial manner with the 2D position and yaw orientation which yields the lowest score.

3.6 Results and Discussion

The proposed approach does not outperform the comparison method on all unaltered test sequences, but it does collectively overcome several of the limitations of traditional methods. As shown in Table 3.1 when evaluated on the unaltered test trajectories, BooM bests VINS-Mono on KITTI for all measures except orientation on sequence 09. On AirSim, VINS-Mono outperforms BooM on translation for both sequences but not orientation.

Results on both KITTI (Tab. 3.2 and Tab. 3.3) and AirSim (Tab. 3.4 and Tab. 3.5) show BooM successfully localizing through all difficulty levels with VINS-Mono failing on many of the



(a) KITTI Rain Results Seq. 09

(b) KITTI Rain Results Seq. 10

Figure 3.7: Top view of the results collected on the KITTI test sequences augmented with rain. VINS-Mono was unable to achieve an accurate initialization in the Heavy and Medium rain conditions on sequences 09 and 10, respectively, and completely unable to initialize in the Heavy rain condition for sequence 10. BooM, however, successfully made it through all conditions with reasonable accuracy (RMSE $\leq$ 22.05 m).

harshest conditions. Figure 3.7 demonstrates these results visually with the rain condition applied to KITTI. On the trajectories VINS-Mono is able to initialize in and not lose tracking, however, it often demonstrates superior translation and orientation performance.

Occasional lapses in accurate egomotion estimates by BooM, however, does not result in catastrophic failure, as was observed by VINS-Mono in several of the harsh conditions. Additionally, our trained deep learning approach produces estimates immediately without a brittle initialization sequence requiring large camera motion. When initialized successfully, though, VINS-Mono exhibited excellent performance across the board, which highlights a primary motivation of this work to leverage the known geometric dynamics employed by model-based solutions outside of ideal conditions where these dynamics are less observable.

Table 3.1: Results on the unaltered KITTI and AirSim test trajectories with the best performances highlighted in **bold**. Top: Absolute translation errors (RMSE) in meters. Bottom: Angular error in yaw in degrees.

Translation	*KITTI*		*AirSim*	
#	*09*	*10*	*1*	*2*
BooM	**18.66**	**17.45**	6.57	7.72
VINS Mono	25.64	21.51	**3.20**	**2.27**
Orientation	*KITTI*		*AirSim*	
#	*09*	*10*	*1*	*2*
BooM	**2.98**	3.41	**2.37**	**1.49**
VINS Mono	4.05	**3.31**	5.24	7.97

Critically, BooM demonstrated the ability to generate reasonable estimates with proper scale on all sequences despite relying on an imperfect training signal from VINS-Mono estimates. And this ability was not lost when evaluated on the harsh conditions.

3.7 Conclusion

Despite using only monocular imagery and IMU measurements without access to scaled depth images, BooM is able to estimate scaled trajectories by bootstrapping learned depth image scale through assistance from a traditional VIO method during network training. BooM outperforms VINS-Mono on several metrics for the unaltered test trajectories on both datasets considered as well as many of the harsh environments evaluated.

Comparison with direct methods on the harsh conditions would have provided additional insight into the advantages of BooM over traditional methods since direct methods relying on photometric measures instead of handcrafted image features are less susceptible to image degradations. However, we were unable to get such methods operating sufficiently well on either the KITTI or AirSim dataset to facilitate meaningful analysis.

Table 3.2: Absolute translation errors (RMSE) in meters on the KITTI dataset for the harsh conditions. Results are computed using the odometry metric from [1] over trajectory segments of lengths 100, 200, 300, 400, 500, 600, 700, 800 m. The × indicates failure to successfully initialize.

KITTI Rain	*Light*		*Medium*		*Heavy*	
#	*09*	*10*	*09*	*10*	*09*	*10*
BooM	20.69	19.63	19.73	**20.87**	**21.39**	**22.05**
VINS Mono	**16.77**	**13.15**	**16.86**	×	×	×
KITTI Fog	*Light*		*Medium*		*Heavy*	
#	*09*	*10*	*09*	*10*	*09*	*10*
BooM	83.87	65.04	83.81	61.55	**79.68**	**56.10**
VINS Mono	**17.54**	**28.78**	**58.24**	**38.08**	×	×
KITTI Snow	*Light*		*Medium*		*Heavy*	
#	*09*	*10*	*09*	*10*	*09*	*10*
BooM	94.12	70.46	95.10	**68.58**	**95.91**	**67.72**
VINS Mono	**20.11**	**33.52**	**33.43**	×	×	×

Table 3.3: Angular error in yaw in degrees on the KITTI dataset for the harsh conditions.

KITTI Rain	*Light*		*Medium*		*Heavy*	
#	*09*	*10*	*09*	*10*	*09*	*10*
BooM	**2.25**	**3.57**	**1.98**	**2.96**	**1.96**	**1.87**
VINS Mono	4.72	3.80	4.76	×	×	×
KITTI Fog	*Light*		*Medium*		*Heavy*	
#	*09*	*10*	*09*	*10*	*09*	*10*
BooM	4.90	**1.43**	6.68	4.55	**10.42**	**9.38**
VINS Mono	**4.72**	6.50	**4.2**	**4.48**	×	×
KITTI Snow	*Light*		*Medium*		*Heavy*	
#	*09*	*10*	*09*	*10*	*09*	*10*
BooM	**3.71**	9.75	**4.02**	**9.97**	**4.20**	**10.45**
VINS Mono	4.71	**4.40**	4.59	×	×	×

Table 3.4: Absolute translation errors (RMSE) in meters on the AirSim dataset for the harsh conditions. Results are computed using the odometry metric from [1] over trajectory segments of lengths 25, 50, 75, 100, 125 m. The × indicates failure to successfully initialize.

AirSim Rain	*Light*		*Medium*		*Heavy*	
#	*1*	*2*	*1*	*2*	*1*	*2*
BooM	18.22	23.87	**19.86**	**24.88**	**21.13**	**25.20**
VINS Mono	**18.20**	**23.28**	×	×	×	×
AirSim Fog	*Light*		*Medium*		*Heavy*	
#	*1*	*2*	*1*	*2*	*1*	*2*
BooM	20.28	**20.96**	37.61	**20.32**	**55.82**	**23.90**
VINS Mono	**7.60**	25.93	**3.18**	×	×	×
AirSim Snow	*Light*		*Medium*		*Heavy*	
#	*1*	*2*	*1*	*2*	*1*	*2*
BooM	14.69	**23.00**	**15.02**	**22.62**	**14.84**	**22.20**
VINS Mono	**11.48**	×	21.65	×	×	×

Table 3.5: Angular error in yaw in degrees on the AirSim dataset for the harsh conditions.

AirSim Rain	*Light*		*Medium*		*Heavy*	
#	*1*	*2*	*1*	*2*	*1*	*2*
BooM	9.03	12.53	**6.34**	**13.15**	**11.05**	**13.42**
VINS Mono	**5.55**	**8.63**	×	×	×	×
AirSim Fog	*Light*		*Medium*		*Heavy*	
#	*1*	*2*	*1*	*2*	*1*	*2*
BooM	8.69	14.49	9.71	**11.32**	**11.13**	**9.75**
VINS Mono	**5.25**	**7.86**	**5.28**	×	×	×
AirSim Snow	*Light*		*Medium*		*Heavy*	
#	*1*	*2*	*1*	*2*	*1*	*2*
BooM	9.35	**16.49**	9.23	**15.82**	**9.25**	**15.63**
VINS Mono	**5.53**	×	**5.50**	×	×	×

BooM did not strictly outperform the traditional VIO method across all metrics, but reasonable results were shown and the primary benefit of BooM's architecture was thoroughly observed with several examples of increased robustness to visual degradations characteristic of common real-world self-localization challenges. BooM's architecture is also shown to be applicable to both ground and aerial vehicles, and presumably any platform with only a monocular camera and IMU.

Estimating Scale with Linear Algebra

We have also explored an additional route to scale estimation by following more closely the methods by which model-based monocular VIO approaches estimate scale with linear algebra. We have, however, employed the same technique without the fragile dependence of finding image features which match across many frames. The traditional approach consists of an initialization sequence of tracking features across frames and using the eight-point [61] or five-point algorithm [5] to estimate unscaled change in poses and combining these results with IMU measurements and camera parameters to solve for scale, gravity, velocity, and bias terms. We, however, do away with the image feature tracking and instead use unscaled change in pose estimates from a monocular VIO network in combination with IMU measurements to acquire scale, gravity, velocity, and bias terms. Figure 3.8 shows how scale can be recovered from an improperly scaled VIO output. Additionally, we incorporate nonlinear least-squares optimization, another model-based tool, to run on the backend and smooth the output poses of a sliding window.

Nonlinear Least-Squares Optimization

Bundle adjustment (BA) improves SLAM approaches through nonlinear least-squares minimization of reprojection errors between detected feature locations and predicted feature locations in images using camera poses, intrinsic camera parameters, and, for the case of visual-inertial SLAM (VI-SLAM), IMU data [62]. In general, nonlinear least-squares optimiziation is the process of minimizing a function

$$F(\mathbf{x}; \Theta) = \sum_{i=1}^{m} r_i(\mathbf{x}; \Theta)^2$$

Here $\mathbf{x}$ is some set input values and Θ is a set of parameters. The quantities r_i (which are functions of $\mathbf{x}$ and Θ) are known as *residuals* – generally each residual some kind of error function (non-linear in the current context) and the task of non-linear least squares optimization is to find

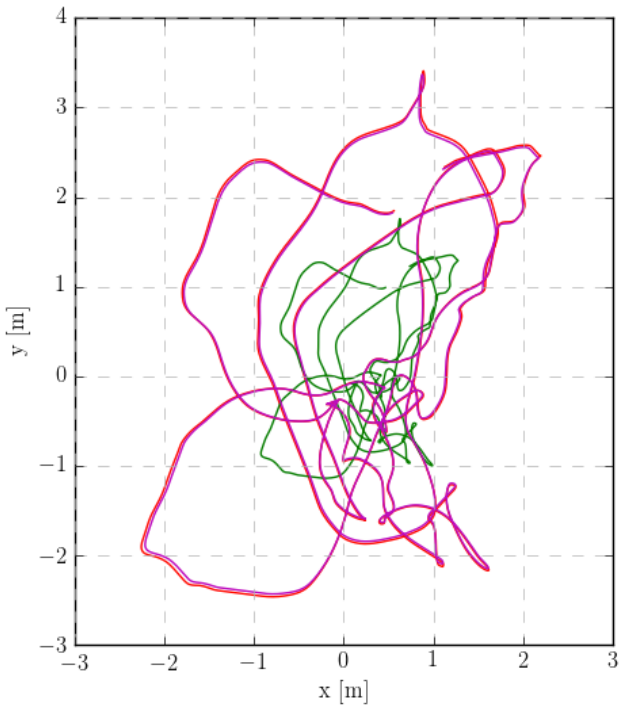


Figure 3.8: Example of scale successfully recovered from model-free monocular VIO unscaled output (green) with the scaled estimate (red) closely matching the ground truth (magenta). Here, the unscaled trajectory is approximately half the scale of the ground truth, and similar scale recovery results have been observed with scale estimates much further from the ground truth, both smaller and larger.

the parameters which minimize the overall error F; i.e. we want to find

$$\Theta^* = \min_{\Theta} F(\mathbf{x}; \Theta)$$

Computational advancements to bundle adjustment (BA) techniques have enabled use with greater time scales for SLAM [63, 64] and VI-SLAM [65], and efficient optimization platforms have emerged. We use Ceres [66] to solve the nonlinear least-squares optimization problem that results in pose output which conforms to the sliding window of previous estimates (to bound complexity).

Chapter 4

MULTI-DOMAIN SELF-LOCALIZATION

Like many current deep learning areas of research, state-of-the-art (SoA) vision-aided localization solutions fail to generalize well outside their target domain. The growing use of cheap, synthetically generated data and domain-adaptation techniques can alleviate this issue for several application areas, but the success of these current methods is limited to less complex vision tasks such as classification and recognition. Further, large domain deviations such as weather conditions and scene motion remains a critical topic. In this work, we present a Mixture of Experts (MoE) solution to the multi-domain application of visual odometry (VO) that retains and often improves expert self-localization performance while also demonstrating robustness to unseen domains. Our end-to-end, unsupervised deep neural network domain experts simultaneously learn to estimate pose and scene depth in their respective domains with several expert combination schemes explored for effective leveraging of expert knowledge. We evaluate a SoA model-based VO approach and our MoE VO solution against networks trained with all available domain data on the KITTI driving dataset as well as a micro aerial vehicle (MAV) dataset we collected in the AirSim simulation environment. Through improved self-localization across multiple weather conditions as well as across vehicle types, we demonstrate a VO framework with the capacity to operate in and benefit from boundless domains.

4.1 Introduction

Despite the introduction of countless advancements over several decades, the well-established robot navigation area of research remains exceedingly popular due to its incredible difficulty and number of unsolved problems. For the subtask of self-localization, many of the challenges can be attributed to photometric complexity and the need to perceive infinitely variable environments.

Some success has been demonstrated by traditional, model-based self-localization techniques as well as neural network-leveraging model-free techniques but these approaches demonstrate their abilities with scenarios closely matching that for which they were finely tuned. We have previously demonstrated improved self-localization robustness with model-based and model-free approaches [67], but further methods are sought for moving towards the goal of a self-localization solution robust to highly diverse domains.

The reliance of model-based methods on handcrafted features to perceive the environment reduces their ability to generalize due to insufficient sensing measures in the presence of novel scenes. The model-free approach with training of a single network similarly faces challenges to cross-domain operation by forever balancing overfitting and generality with the need to obtain more diverse training data. Even with access to abundant training samples, however, training with data from multiple diverse domains does not guarantee improved generality with refined accuracy. With excessive data and an excessive amount to learn, the typical network architecture must delicately manage learned knowledge to avoid overwriting and relearning previous knowledge when dealing with unbalanced domain samples.

Transferring learned knowledge from one specialized domain to another has shown some promise with domain adaptation techniques, but no current solution exists for complex vision tasks and consideration of additional diverse domains falls outside the scope. Ensemble methods, however, present a suitable means for addressing cross-domain VO with the capability of preserving and improving domain-specific knowledge through a process of splitting a large problem into sub-problems to be solved by independent learners and intelligently combining their results to maximizing task accuracy.

In this work, we present an end-to-end trainable monocular VO approach which effectively combines pose estimates from independent modules trained for a unique domain (Fig. 4.1). To the best of the authors' knowledge, this work is the first to

1. Utilize a Mixture of Experts (MoE) framework to achieve improved unsupervised visual odometry (VO) estimation from a combination of independent estimators

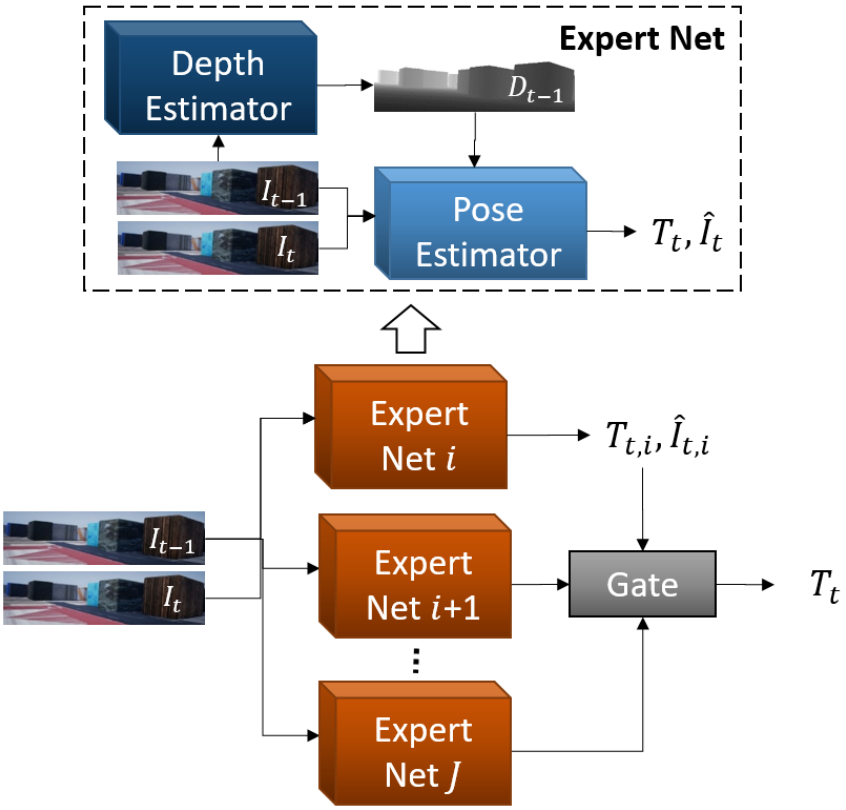


Figure 4.1: Overview of the MoE VO architecture. Expert VO estimators consist of a model-free pose estimator which takes RGB images from the source and target time instances, I_{t-1}, I_t, and an estimated depth image at the source time, D_{t-1}, generated from a model-free depth estimator to produce a pose estimate. Additionally, experts output a reconstructed target image, $\hat{I}_t$, used by the MoE gate to assign each of the J expert contributions to the final estimate, T_t.

2. Demonstrate VO estimation across diverse visual and motion domains

4.2 Approach

MoE VO is an unsupervised monocular VO deep network that estimates the ego-motion of a moving camera between some time t_{j-1} and time t_j, the two most recent images captured by the camera, from a combination of refined VO experts. Experts receive two RGB images (I_{j-1}, I_j) and the camera's intrinsic parameters (K matrix) as their only input. With access to K, MoE VO can generate camera pose changes in the camera frame using a view-synthesis approach where the basis for its training is in the Euclidean loss between a target image and a reconstructed target image generated using pixels in a source image sampled at locations determined by a learned 3D affine transformation (via a spatial transformer of [46]). The 3D affine transformation is facilitated by an integrated depth network based upon SFMLearner [20] which learns to produce a depth image from the input monocular imagery (Fig. 4.2).

MoE VO's expert networks are trained for a unique domain with the ensemble architecture following that of the typical MoE scheme of utilizing a divide-and-conquer strategy and combining experts with a gate module. MoE VO differs, however, in that each individual expert learner solves the same problem while specializing in a unique range of image and motion profiles. More formally, assuming J experts with output y and input x, each expert h_i produces an estimate of y from $h_i(y|x;\theta_i)$, where θ_i is the parameters of the ith expert. The gate applies coefficients $\pi_i(x;\alpha)$ that weigh expert contributions, with α being the gating function parameters (Fig 4.1). Together, the MoE method is expressed by

$$H(y|x;\Psi) = \sum_{i=1}^{J}(\pi_i(x;\alpha) \cdot h_i(y|x;\theta_i)$$

where Ψ represents the unknown parameters [42]. Additionally, the gating function typically utilizes the softmax function to apply probability weights to the experts.

The MoE gate serves a critical role in the overall performance of the ensemble method, tasked with combining expert estimations most effectively at each frame. A neural network may provide

Figure 4.2: Sample RGB images and learned depth counterparts in the AirSim MAV dataset (top row) and the KITTI driving dataset (bottom row).

additional insight into expert combination with CNNs proven to extract meaningful information from complex data, particularly imagery. As such, we explore training a gate function to serve as a pose error estimator with the goal to outperform direct measures of expert pose accuracy.

In our unsupervised framework, we lack a perfect assessment of pose estimations made by experts and therefore have to estimate their accuracy to construct a gate that assigns a large contribution from the top-performing expert and a small contribution from other experts. VIOLearner, along with many other unsupervised self-localization networks, reliably uses image reprojection errors as a loss signal to measure the quality of motion estimation and guide learning, popularized by [68]. As mentioned in 1.1.6, a view-synthesis approach facilitates learning by generating a novel view from a source RGB and depth image. Taking an estimated change in pose, the camera intrinsics matrix, and the depth image, source RGB pixels are projected forward, estimating the camera view at the new pose. With access to the next frame showing the actual camera view at the new pose, an error image can be computed in numerous ways to assess pose estimation.

4.3 Methods

4.3.1 Hyperparameter Optimization

A thorough comparison between the MoE solution and a single network solution trained with access to all available domain data is critical for the analysis of the proposed approach. This presents a challenge, however, because whether the MoE solution or the single network solution generates better results may occur simply from not training the networks as best as possible.

To address this concern, hyperparameter optimization is employed to fully and objectively sample the hyperparameters that guide learning. The Tree of Parzen Estimators (TPE) is used with the Hyperopt implementation [69] to optimize the learning rate, refinement weight on convolutional layers, reprojection loss weight, depth smoothness weight, depth structural similarity weight, and the minimum disparity for the depth estimation subnetwork.

4.3.2 Training Procedures

All of the trained networks underwent similar training procedures following their hyperparameter search procedure. Each was trained using a batch size of 16 and allowed to train for at most $200,000$ iterations with an evaluation occurring every $1,000$ iterations on the respective validation set. We used the Adam solver with momentum1=0.9, momentum2=0.99, gamma=0.5, and an exponential staircase learning rate policy. The learning rate decreased by a factor of 0.3 every $25,000$ iterations. The networks were trained using single-point precision (FP32) on desktop computers with 2.60 GHz Intel i9-7980XE processors and Nvidia GPUs.

4.3.3 KITTI Driving Dataset

We have elected to use the popular KITTI odometry dataset [51] to evaluate our approach on a ground vehicle platform. The KITTI dataset includes RGB and depth imagery captured at 10 Hz as well as IMU data captured at 100 Hz over a driving distance of $39.2\,\mathrm{km}$ around a residential area. All 10 trajectories were groundtruthed with accuracies within $10\,\mathrm{cm}$ via an OXTS RT 3003

GPS solution. For this VO work, we exclude the available depth and IMU input modalities to reduce sensor requirements and evaluate the proposed method with reduced sensor variation effects. Additionally, the raw RGB imagery is downsampled from $376x1241$ to $128x480$ resolution.

4.3.4 AirSim MAV Simulation Environment

An experimental environment (Fig. 3.5) built within Unreal Engine, a game development platform that enables realistic, high-fidelity visualizations and physics simulations, was used to support the creation of a micro aerial vehicle (MAV) dataset. Aerial vehicle control leverages the Microsoft AirSim plugin [52] for simulated flight dynamics as well as a suite of sensors including inertial (GPS, barometer, gyroscope, accelerometer) and imaging (RGB, depth, segmentation). Absolute groundtruth is provided by the virtual environment, facilitating near-perfect error measures. For this work, 128x480 resolution monocular RGB imagery is captured at 11.11 Hz.

Manual flight control through the environment, which resembles a city block, provided turn and velocity inputs for each trajectory, with AirSim managing drone dynamics such that turns and changes in velocity apply realistic effects to roll, pitch, and yaw motion. Velocity was kept between 0 m/s and 8 m/s during all flights. A total of 13 trajectories were captured within the environment, with each trajectory containing 126 seconds of flight time. The dataset was split with 10 of the trajectories in the training set, 1 in the validation set, and 2 in the test set.

4.3.5 Augmenting Datasets with Diverse Weather Conditions

The flight commands for each trajectory in the AirSim synthetic dataset were recorded for efficient and consistent playback to enable additional versions of each trajectory that differ only in their observed imagery. Doing so facilitates visual odometry (VO) analysis that elicits differences solely due to photometric variation without effects from motion variation. Two additional versions of the AirSim dataset were collected, with the first representing the baseline *Sun* weather condition, followed by *Rain* and *Snow* replicas, shown in Fig. 4.3. For the *Rain* environment, the sky was darkened with a haze added within the Unreal Engine. Post-processing with the Adobe After

Figure 4.3: Sample images from the AirSim weather conditions trajectories showing the *Sun*, *Rain*, and *Snow* effects.

Effects software added falling rain to these trajectories which considers the depth of the scene to improve realism. Similarly, the *Snow* environment underwent a sky and haze modification in the Unreal Engine with the Adobe After Effects software facilitating the addition of falling snow. The *Snow* imagery, however, includes added difficulty in the form of a texture-sparse snow-covered ground.

4.4 Evaluation

Evaluation is conducted on both the KITTI driving and AirSim MAV weather condition datasets against networks trained with all domain data. The well-established ORB-SLAM [7] algorithm,

representing a SoA model-based VO method, is also evaluated on the AirSim weather condition test trajectories to benchmark the difficulty of tracking in these harsh environments. For fair comparison as a pure VO method, ORB-SLAM's relocalization and loop closure modules are disabled.

We follow the popular KITTI error metrics [51] for effective Visual Odometry evaluation which builds upon the work of [70] for computing relative error with respect to fixed distances, with translation and rotation errors considered separately. These metrics provide a more encompassing measure compared to end-point evaluation, which can be heavily varied based on the distribution of rotation errors along the length of interest. For all AirSim trajectories, $t_r(\%)$ is the average relative translational error percentage and $r_r(^\circ)$ is the average relative rotational error $(^\circ/100\,\mathrm{m})$ computed on all lengths $25\,\mathrm{m}$ - $125\,\mathrm{m}$ in $25\,\mathrm{m}$ increments. For all KITTI trajectories, $t_r(\%)$ is the average relative translational error percentage and $r_r(^\circ)$ is the average relative rotational error $(^\circ/100\,\mathrm{m})$ computed on all lengths $100\,\mathrm{m}$ - $800\,\mathrm{m}$ in $100\,\mathrm{m}$ increments.

Lastly, for evaluation across unscaled VO trajectory estimates, we optimize pose estimate scale to best align with the groundtruth.

4.5 Results and Discussion

4.5.1 Image Similarity

Using the reconstructed target image computed from each expert for gauging pose estimation accuracy promotes exploration of a number of image similarity metrics. Quantitatively measuring image similarity typically consists of calculating the L1 or L2 norm, without strong reasoning for choosing one over the other. It has been shown, however, that human visual perception of image similarity more closely aligns with the L1 metric [71], likely because humans weight pixel differences equally in the manner of calculating the L1 norm whereas the L2 norm squares the difference, penalizing unequally. Formally, the calculation of the L1 norm follows:

$$\mathcal{L}_{L1}(P) = \frac{1}{N} \sum_{p \in P} |x(p) - y(p)|$$

between images x and y for a patch P with N pixels where p represents pixel indices. The L2 norm is similarly calculated by:

$$\mathcal{L}_{L2}(P) = \frac{1}{N}\sum_{p \in P} |x(p) - y(p)|^2$$

Image similarity metrics have also evolved from strict pixel-level comparison into a class of metrics, lead by the structural similarity metric (SSIM) [72], interested in comparing structural information. The SSIM metric is calculated by:

$$\mathcal{L}_{SSIM}(P) = \frac{2\mu_{x(P)}\mu_{y(P)} + C_1}{\mu_{x(P)}^2 + \mu_{y(P)}^2 + C_1} \cdot \frac{2\sigma_{x(P)y(P)} + C_2}{\sigma_{x(P)}^2 + \sigma_{y(P)}^2 + C_2}$$

where $\mu_{x(P)}$ is the average of x, $\mu_{y(P)}$ the average of y, $\sigma_{x(P)}^2$ the variance of x, $\sigma_{y(P)}^2$ the variance of y, $\sigma_{x(P)y(P)}$ the correlation coefficient of x and y, all over patch P, and C_1 and C_2 are small constants to stabilize division with a weak denominator. The SSIM range of output assigns low values for dissimilarity and 1.0 for two completely identical images. Therefore, we subtract SSIM values from 1.0 to follow $L1$ and $L2$ metrics of associating low values with similarity. For this work, we set the patch size as $3x3$ when computing the SSIM error.

With the proven ability of neural networks to extract meaningful information from complex data, it was theorized a learned error measure could provide deeper insight, leading to intelligent combinations of expert contributions using knowledge beyond what could be collected from the source and target frames at any one instance. The learned gate design takes as input the reconstructed target image from an expert, $\hat{I}_i$, subtracts the real target image, I_t, and runs the computed error image through 6 convolutional layers followed by a single fully-connected layer with a linear activation output function. The SSIM score served as the training signal during training, with training never resulting in a solution that tracked the SSIM precisely. The learned gate here estimates image similarity error alone, and is amenable with any number of experts in the MoE scheme, as is the other non-learned gates.

For the results reported here, the image similarity metric used with the MoE framework is

Table 4.1: AirSim expert results on the AirSim test trajectories from each weather condition.

		Sun		*Rain*		*Snow*	
Seq							
		$t_r(\%)$	$r_r(^\circ)$	$t_r(\%)$	$r_r(^\circ)$	$t_r(\%)$	$r_r(^\circ)$
Sun	1	3.91	10.28	6.06	14.75	5.52	14.23
	2	4.42	11.54	6.78	17.98	8.00	21.06
Rain	1	5.09	11.85	4.65	12.90	5.95	14.94
	2	5.47	12.70	5.45	14.92	7.66	19.52
Snow	1	6.32	14.14	6.58	18.42	4.99	11.93
	2	6.86	14.14	7.24	19.20	5.66	13.83

specified in the title of the tested method, with all other parameters equal. We denote the learned gate method as *MoE CNN* and exclude $L1$ results as these scores did not significantly contribute beyond $L2$ results.

4.5.2 Gate Weighting

All image similarity metrics apply a score to image pairs, providing not just which pair is more similar, but a relative gauge of how much similar. This conceptually leads to a potential gate weighting scheme which assigns contribution as a function of the total error observed from all experts. The gate function for such a strategy would follow:

$$\pi_i(x;\alpha) = \frac{e_{tot}}{e_i e_{tot\_inv}}$$

with

$$e_{tot} = \sum_{i=1}^{J} e_i, \;\; e_{tot\_inv} = \sum_{i=1}^{J} \frac{1}{e_i}$$

for expert i with error e_i. For a simple two expert case with errors 8 and 2, this gate would compute weights of 0.2 and 0.8, respectively.

In practice, however, weighting in such a way often resulted in poor performance, despite assigning little contribution to experts with a high error. A winner-take-all weighting scheme proved much more effective by assigning all contribution for a given frame to the expert with the lowest image similarity error. As such, the rest of this work uses a winner-take-all, 1-hot gate distributor with all image similarity metrics explored.

4.5.3 Visual Domain Variation

To assess the proposed framework with a range of differing dataset domains, we first looked at domains which differed only in their visual appearance of the environment. The AirSim simulation environment was used to create three datasets with similar flight motion but differing visual scenes (Fig. 4.3). A weather domain expert weas trained for each of the weather condition datasets (Tab. 4.1) along with a comparison network trained with the AirSim *Sun* and *Rain* data as well as a network trained with the AirSim *Sun* and *Snow* data. Experiments were performed for two scenarios where data from only two of the three weather conditions were available to determine which approach more intelligently combines samples from multiple domains on seen and unseen domains. The *Sun-Rain* and *Sun-Snow* combinations were chosen to serve as scenarios with two different magnitudes of domain difference. Tables 4.2 and 4.4 contain odometry results for the AirSim *Sun-Rain* and AirSim *Sun-Snow* experiments, respectively. Additionally, Tables 4.3 and 4.5 summarize the MoE gate weight distributions for each of the test sequences, showing the percentage of expert utilization, with a visual example shown in Fig. 4.4.

Sun-Rain

As shown in Tab. 4.2, each MoE gate scheme demonstrates more accurate odometry estimation for all metrics and sequences. The MoE SSIM gate also outperforms or nearly outperforms the other gate combination methods for all sequences, making a case for the importance of perceptual measures and recognizing structure in a scene for the image similarity measure of odometry accuracy with the view synthesis approach. The intended goal of the SSIM score to assess similarity

Table 4.2: AirSim *Sun-Rain* odometry results on the AirSim test trajectories from each weather condition. The best and second best scores are highlighted in red and blue, respectively.

		Sun-Rain Net		*MoE L2*		*MoE SSIM*		*MoE CNN*	
Seq									
		$t_r(\%)$	$r_r(^\circ)$	$t_r(\%)$	$r_r(^\circ)$	$t_r(\%)$	$r_r(^\circ)$	$t_r(\%)$	$r_r(^\circ)$
Sun	1	4.94	13.30	3.62	8.56	3.41	8.78	3.78	9.61
	2	5.76	15.03	4.57	11.92	4.45	11.45	4.67	12.16
Rain	1	4.64	12.96	3.82	10.44	3.68	10.03	4.07	10.40
	2	5.63	14.26	4.93	13.67	4.80	13.03	4.84	13.21
Snow	1	6.10	17.37	5.06	12.44	4.86	11.75	5.34	12.84
	2	6.55	17.03	6.02	15.53	5.76	14.88	6.14	14.68

Table 4.3: AirSim *Sun-Rain* MoE gate distribution results averaged for each weather condition of the AirSim test trajectories. Values shown represent expert contribution as a percentage of the test trajectory.

	MoE L2		*MoE SSIM*		*MoE CNN*	
Cond						
	Sun	Rain	Sun	Rain	Sun	Rain
Sun	69.17	30.83	76.36	23.64	69.92	30.08
Rain	23.64	76.36	31.90	68.10	41.81	58.19
Snow	43.38	56.62	47.00	53.00	51.29	48.71

Table 4.4: AirSim *Sun-Snow* odometry results on the AirSim test trajectories from each weather condition. The best and second best scores are highlighted in red and blue, respectively.

		Sun-Snow Net		*MoE L2*		*MoE SSIM*		*MoE CNN*	
Seq									
		$t_r(\%)$	$r_r(^\circ)$	$t_r(\%)$	$r_r(^\circ)$	$t_r(\%)$	$r_r(^\circ)$	$t_r(\%)$	$r_r(^\circ)$
Sun	1	4.04	9.98	3.57	9.14	3.47	8.90	3.95	9.62
	2	5.25	13.51	4.28	11.42	4.35	11.41	4.43	12.06
Rain	1	4.50	11.69	4.59	10.87	4.38	9.79	4.84	11.01
	2	5.61	14.26	5.17	12.99	5.14	12.84	5.53	13.63
Snow	1	4.52	10.29	4.82	10.67	4.94	10.86	4.82	10.63
	2	5.51	13.79	5.23	12.22	5.31	12.07	5.70	13.07

Table 4.5: AirSim *Sun-Snow* MoE gate distribution results averaged for each weather condition of the AirSim test trajectories. Values shown represent expert contribution as a percentage of the test trajectory.

	MoE L2		*MoE SSIM*		*MoE CNN*	
Cond						
	Sun	Snow	Sun	Snow	Sun	Snow
Sun	82.69	17.31	83.80	16.20	75.57	24.43
Rain	65.20	34.80	72.35	27.65	64.20	35.80
Snow	28.00	72.00	33.33	66.67	46.14	53.86

beyond individual pixel differences may also serve the proposed method well due to the relatively low resolution of imagery used and the commonly observed pixel-level artifacts that may cause issues with absolute measures such as $L2$.

Sun-Snow

Similarly, the MoE SSIM gate outperforms or nearly outperforms the other gate combination methods for all sequences except for one of the *Snow* sequences (Tab. 4.4). The single *Sun-Snow* network may exhibit greater performance here with a lack of *Rain* data, which proves challenging for the network to learn and negatively impacts learning of the other domains, as shown by greater *Sun* accuracy by the *Sun-Snow* network in Tab. 4.4 compared to the *Sun* accuracy of the *Sun-Rain* network in Tab. 4.2. Further, the *Sun* expert demonstrated comparable performance to the *Rain* expert on the *Rain* test trajectories (Fig. 4.1). This is likely attributed to poor depth estimation performance in the *Rain* environment due to the density of the falling rain causing sporadic holes in the estimated depth image.

Less significant odometry improvements with the MoE in this *Sun-Snow* experiment compared to the *Sun-Rain* experiment may also be explained by the greater diversity of these two domains. The gate distribution tables (Tab. 4.3 and 4.5) show less balanced expert utilization by the MoE framework when using the *Sun* and *Snow* experts, communicating lesser abilities to assist another expert when operating in that expert's domain.

4.5.4 Visual + Motion Domain Variation

We then extended experimentation to assess the MoE framework with a set of domain experts with even greater diversity. The combined KITTI and AirSim experiment odometry results are shown in Tab. 4.6. A comparison network trained with equal parts AirSim *Snow* and KITTI data, AS-K$_B$, serves to further analyze the ability of a single VO network to simultaneously learn accurate odometry estimation for two domains with large visual and motion diversity. Approximately one-fourth of the KITTI data is excluded to match the number of AirSim *Snow* samples. Additionally,

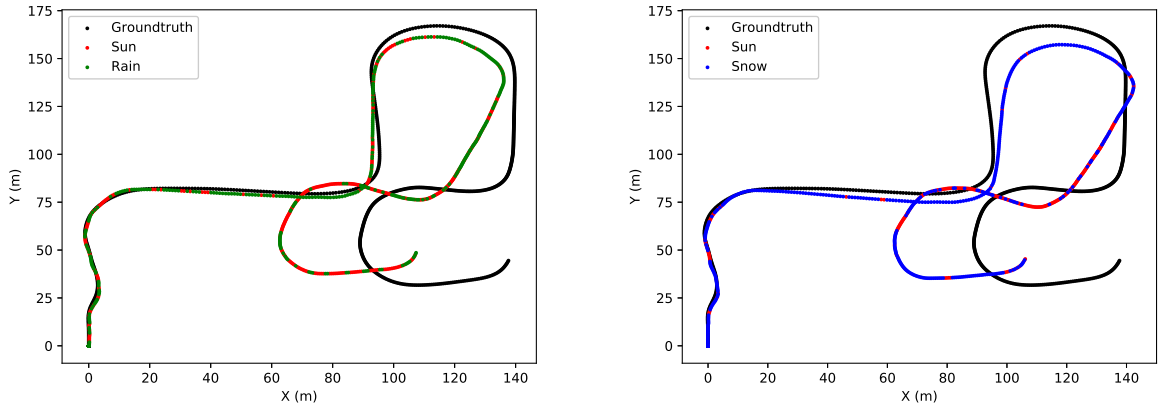


Figure 4.4: Trajectories showing pose estimation expert usage dictated by the MoE SSIM gate. The AirSim *Sun-Rain* net shows a balanced usage on a snow test trajectory (left) while the AirSim *Sun-Snow* net (right) shows a much greater reliance on the *Snow* expert estimates on the same test trajectory.

AS-K$_U$ uses all of the KITTI training data and serves to analyze potential skewing effects of a solution trained with an unbalanced set of domain samples.

Looking first at the AirSim *Snow* results from the AS-K$_B$ and AS-K$_U$ networks, both translation and orientation performance on the *Snow* test trajectories degrades when the network contains a disproportionate amount of domain training samples. Orientation error does, however, decrease for one of the test trajectories compared to the *Snow* expert for both networks. Results on the KITTI test set shows the AS-K$_U$ network demonstrating greater KITTI accuracy compared to the AS-K$_B$ network, which is likely attributed to the unbalanced learning issue.

None of the MoE methods are able to gain sizeable accuracy gains beyond the *Snow* expert's estimates, likely due to the relatively poor results from the KITTI expert causing both the $L2$ and SSIM gate schemes to not make notable useage of the KITTI expert's estimates, as shown in Tab. 4.7. A more relevant expert would have likely increased performance, as previously shown with the experiments only considering visual variation, though both the $L2$ and SSIM gate schemes here are notably able to retain the *Snow* expert's results without much degradation while also considering the KITTI expert.

Table 4.6: Odometry results on the AirSim *Snow* and KITTI test trajectories. *AS Snow** represents the *Snow* expert trained with approximatly 10% KITTI data, AS-K_B represents the single network trained with an equal ratio of AirSim *Snow* training samples and KITTI training samples, with approximately one fourth of the KITTI data is excluded, and AS-K_U represents the single network trained with all the AirSim *Snow* training data as well as all of the KITTI training data. The best and second best scores are highlighted in red and blue, respectively.

		*AS Snow**		*KITTI*		*AS-K_B*		*AS-K_U*		*MoE L2*		*MoE SSIM*		*MoE CNN*	
Seq															
		$t_r(\%)$	$r_r(^\circ)$	$t_r(\%)$	$r_r(^\circ)$	$t_r(\%)$	$r_r(^\circ)$	$t_r(\%)$	$r_r(^\circ)$	$t_r(\%)$	$r_r(^\circ)$	$t_r(\%)$	$r_r(^\circ)$	$t_r(\%)$	$r_r(^\circ)$
AS Snow	1	3.95	9.85	25.59	51.74	4.44	10.48	5.46	13.27	3.95	9.85	3.95	9.85	4.11	10.23
	2	4.60	12.28	27.58	56.55	4.72	11.25	5.44	11.82	4.59	12.26	4.65	12.28	5.48	13.35
KITTI	1	50.23	18.52	5.38	2.76	6.91	3.00	5.89	3.10	5.38	2.76	5.29	2.75	4.98	2.74
	2	35.38	20.05	6.37	3.98	8.27	4.59	7.82	3.46	6.37	3.98	6.41	3.85	6.46	3.94

These results show the AirSim *Snow* and KITTI domains differ too greatly for effective MoE combinations, with the KITTI expert providing few contributions in the MoE scheme. As such, we incorporated KITTI training samples into the AirSim *Snow* training dataset to create a new AirSim *Snow** expert trained with approximately 10% KITTI data. We find this small amount indeed boosts generality and improves its usefulness in the MoE framework and relevance when evaluating on the KITTI test trajectories. Tab. 4.7 shows greater usage of the *Snow* expert by two of the gate schemes and some improvement in performance compared to the KITTI expert (Tab. 4.6).

In contrast to the *Snow** expert, which achieves greater odometry accuracy on the *Snow* test trajectories with the addition of 10% of KITTI training data, the mixed datasets, AS-K_B and AS-K_U, exhibit only a few minor rotation improvements over the *Snow** and KITTI experts on their respective domains. The complexity of the pose estimation problem may provide some reasoning for this phenomenon as too much training data diversity degrades solution precision whereas some diversity serves to reduce overfitting.

Table 4.7: AirSim *Snow* and KITTI MoE gate distribution results averaged for the AirSim *Snow* and KITTI test trajectories. Values shown represent expert contribution as a percentage of the test trajectory.

	MoE L2		*MoE SSIM*		*MoE CNN*	
Cond						
	Snow	KITTI	Snow	KITTI	Snow	KITTI
Snow	100.00	0.00	100.00	0.00	96.67	3.33
KITTI	0.00	100.00	2.83	97.17	12.04	87.96

4.5.5 Harsh Conditions with Model-Based VO

Odometry evaluation on the AirSim *Sun*, *Rain*, and *Snow* test trajectories with a pure VO version of ORB-SLAM demonstrate the shortcomings of using model-based handcrafted features, particularly with the goal of achieving robustness to unseen domains. *Sun* testing produced relative translational scores of 21.90% and 22.35%, and relative rotational scores of 8.08° and 4.47° for the respective test trajectories. The density and rapid movement of the falling rain, however, seemed to interfere too greatly with feature tracking, impeding ORB-SLAM initialization. While initialization was often achieved in the *Snow* environment, so was lost tracking and severe degradation of motion estimation likely caused by feature-sparse frames. Extensive parameter tuning of ORB-SLAM's feature parameters only resulted in poor translation and decent rotation results in the ideal *Sun* weather condition and failed tracking in the *Rain* and *Snow* conditions.

4.6 Conclusion

In this work, we have presented a multi-domain self-localization framework and demonstrated superior performance over domain experts and single network solutions when operating in both visually and movement diverse domains. A MoE approach with image reconstruction error gate weighting schemes combine knowledge from diverse domains better than the comparison methods for nearly every trajectory assessed, though the effectiveness of improving estimates drops slightly

with excessively diverse domains, as found with the AirSim *Snow* and KITTI tests. The learned error metric (MoE CNN) showed significant abilities for intelligently combining experts in the AirSim *Snow* and KITTI tests and warrants additional exploration while the SSIM error metric proved useful across almost all expert combining experiments.

Further, the use of an untrained gate scheme facilitates the addition and removal of odometry experts without configuration changes. Odometry estimation also appears to often grow with the inclusion of additional domain experts, with computational resources being the only limiting factor. The direct measuring of gate weights, however, provides a straightforward means of helping address the computational demands of a multi-expert solution by tracking expert usage over a recent window of estimates and selectively turning off domain experts when usage falls below a threshold.

Lastly, the framework is independent of the neural network used here, though it does rely on the commonly used reconstructed target image to assign gate weights. Classical model-based self-localizers can also integrate well if reconfigured to project source images forward using pose estimates, though we have shown these methods lack robustness to unseen domains.

4.7 Future Work

The experiments conducted with the MoE framework were assessing pure monocular VO for systems using only a monocular RGB camera. The simple setup was sufficient for the purposes of the methods explored in this chapter but a deployable self-localization system needs to consider several additional mechanisms to be useful. To remain limited in hardware requirements, it would be worthwhile to include IMU input and produce scaled estimates using the work from Chapter 3. Even with absolute scale estimated, however, without the ability to sync pose estimates with known points in the environment, trajectory drift appears over time as small pose update errors accumulate. An often effective solution to this problem utilizes previously visited locations to align pose estimates made between a repeat visit to a particular location, effectively closing the loop. Loop closure consists of two primary operations: place recognition to identify repeat visits and pose graph optimization. Pose graph optimization estimates relative pose estimates, using loop

closing poses as constants while minimizing position and orientation residuals of the connecting poses.

Combining domain experts fails to serve a resource constrained system well. Network pruning appears as a viable route to help address this issue, as it has been shown to reduce memory and energy consumption while also increasing generality of trained solutions [73]. The process of network pruning involves eliminating connections in a trained network which have weights that fall below a certain threshold, as these connections effectively contribute little to the output. The hypothesis given for why such occurrences arise with a trained network performing well without fully utilizing all connections is that the initialization of only a subset of the weights resulted in meaningful correlations of the input. Additionally, the smaller subnetwork may be less susceptible to overfitting, following the conventional view that less complex models can better generalize to unseen (test) data.

BIBLIOGRAPHY

[1] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.

[2] F. Fraundorfer and D. Scaramuzza, "Visual odometry: Part ii: Matching, robustness, optimization, and applications," *IEEE Robotics & Automation Magazine*, vol. 19, no. 2, pp. 78–90, 2012.

[3] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the Mars Exploration Rovers," *Journal of Field Robotics*, vol. 24, no. 3, pp. 169–186, 2007.

[4] H. P. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover." STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, Tech. Rep., 1980.

[5] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *CVPR*, vol. 1. Ieee, 2004, pp. I–I.

[6] F. Fraundorfer and D. Scaramuzza, "Visual odometry: Part ii - matching, robustness, and applications," *IEEE Robotics & Automation Magazine*, vol. 19, pp. 78–90, 06 2012.

[7] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[8] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *ICRA*. IEEE, 2007, pp. 3565–3572.

[9] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback," *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1053–1072, 2017.

[10] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[11] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[12] M. Li and A. I. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

[13] M. N. Galfond, "Visual-inertial odometry with depth sensing using a multi-state constraint Kalman filter," Master's thesis, Massachusetts Institute of Technology, 2014.

[14] F. Pang, Z. Chen, L. Pu, and T. Wang, "Depth enhanced visual-inertial odometry based on multi-state constraint Kalman filter," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 1761–1767.

[15] J. Zhang and S. Singh, "Enabling aggressive motion estimation at low-drift and accurate mapping in real-time," in *ICRA*. IEEE, 2017, pp. 5051–5058.

[16] R. Garg, V. K. BG, G. Carneiro, and I. Reid, "Unsupervised CNN for single view depth estimation: Geometry to the rescue," in *ECCV*. Springer, 2016, pp. 740–756.

[17] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *CVPR*, vol. 2, no. 6, 2017, p. 7.

[18] H. Zhan, R. Garg, C. S. Weerasekera, K. Li, H. Agarwal, and I. Reid, "Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction," in *CVPR*. IEEE, 2018, pp. 340–349.

[19] S. Pillai and J. J. Leonard, "Towards visual ego-motion learning in robots," *arXiv preprint arXiv:1705.10279*, 2017.

[20] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *CVPR*, vol. 2, no. 6, 2017, p. 7.

[21] R. Mahjourian, M. Wicke, and A. Angelova, "Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5667–5675.

[22] R. Li, S. Wang, Z. Long, and D. Gu, "UnDeepVO: Monocular visual odometry through unsupervised deep learning," *arXiv preprint arXiv:1709.06841*, 2017.

[23] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni, "VINet: Visual-inertial odometry as a sequence-to-sequence learning problem." in *AAAI*, 2017, pp. 3995–4001.

[24] C. Chen, S. Rosa, Y. Miao, C. X. Lu, W. Wu, A. Markham, and N. Trigoni, "Selective sensor fusion for neural visual-inertial odometry," *arXiv preprint arXiv:1903.01534*, 2019.

[25] S. Choi, J. H. Joung, W. Yu, and J.-I. Cho, "What does ground tell us? monocular visual odometry under planar motion constraint," in *2011 11th International Conference on Control, Automation and Systems*. IEEE, 2011, pp. 1480–1485.

[26] D. Scaramuzza, F. Fraundorfer, and R. Siegwart, "Real-time monocular visual odometry for on-road vehicles with 1-point ransac," in *2009 IEEE International Conference on Robotics and Automation*. Ieee, 2009, pp. 4293–4299.

[27] B. Lee, K. Daniilidis, and D. D. Lee, "Online self-supervised monocular visual odometry for ground vehicles," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5232–5238.

[28] S. Song, M. Chandraker, and C. C. Guest, "High accuracy monocular sfm and scale correction for autonomous driving," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 4, pp. 730–743, 2016.

[29] X. Yin, X. Wang, X. Du, and Q. Chen, "Scale recovery for monocular visual odometry using depth estimated with deep convolutional neural fields," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5870–5878.

[30] V. Peretroukhin and J. Kelly, "Dpc-net: Deep pose correction for visual localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2424–2431, 2018.

[31] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[32] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," *CoRR*, vol. abs/1605.06457, 2016. [Online]. Available: http://arxiv.org/abs/1605.06457

[33] C.-J. Tsai, Y.-W. Tsai, S.-L. Hsu, and Y.-C. Wu, "Synthetic training of deep cnn for 3d hand gesture identification," in *Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO), 2017 International Conference on*. IEEE, 2017, pp. 165–170.

[34] L. Xie, S. Wang, A. Markham, and N. Trigoni, "Towards monocular vision based obstacle avoidance through deep reinforcement learning," *arXiv preprint arXiv:1706.09829*, 2017.

[35] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4293–4302.

[36] Y. Mou, M. Gong, H. Fu, K. Batmanghelich, K. Zhang, and D. Tao, "Learning depth from monocular videos using synthetic data: A temporally-consistent domain adaptation approach," *arXiv preprint arXiv:1907.06882*, 2019.

[37] F. Engelmann, T. Kontogianni, A. Hermans, and B. Leibe, "Exploring spatial context for 3d semantic segmentation of point clouds," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 716–724.

[38] J. Lee, S. Hwang, K. Lee, W. J. Kim, J. Lee, T.-y. Chung, and S. Lee, "Ad-vo: Scale-resilient visual odometry using attentive disparity map," *arXiv preprint arXiv:2001.02090*, 2020.

[39] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE transactions on pattern analysis and machine intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[40] Y. Freung and R. Shapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J Comput Syst Sci*, vol. 55, pp. 119–139, 1997.

[41] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[42] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012.

[43] C. Herdtweck and C. Curio, "Experts of probabilistic flow subspaces for robust monocular odometry in urban areas," in *2012 IEEE Intelligent Vehicles Symposium*. IEEE, 2012, pp. 661–667.

[44] A. Valada, J. Vertens, A. Dhall, and W. Burgard, "Adapnet: Adaptive semantic segmentation in adverse environmental conditions," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4644–4651.

[45] E. J. Shamwell, S. Leung, and W. D. Nothwang, "Vision-aided absolute trajectory estimation using an unsupervised deep network with online error correction," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[46] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial Transformer Networks," *Nips*, pp. 1–14, 2015.

[47] P. Anandan, J. Bergen, K. Hanna, and R. Hingorani, "Hierarchical model-based motion estimation," in *Motion Analysis and Image Sequence Processing*. Springer, 1993, pp. 1–22.

[48] E. J. Shamwell, W. D. Nothwang, and D. Perlis, "An embodied multi-sensor fusion approach to visual motion estimation using unsupervised deep networks," *Sensors (Basel, Switzerland)*, vol. 18, no. 5, 2018.

[49] J. Wang and L. Perez, "The effectiveness of data augmentation in image classification using deep learning," Technical report, Tech. Rep., 2017.

[50] A. Handa, V. Patraucean, V. Badrinarayanan, S. Stent, and R. Cipolla, "Understanding real world indoor scenes with synthetic data," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[51] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[52] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and service robotics*. Springer, 2018, pp. 621–635.

[53] R. Resnick and D. Halliday, *Physics*, ser. Physics. Wiley, 1966, no. pt. 1. [Online]. Available: https://books.google.com/books?id=JncgAQAAIAAJ

[54] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," *Memory*, vol. 10, p. 20, 2018.

[55] E. J. Shamwell, K. Lindgren, S. Leung, and W. D. Nothwang, "Unsupervised deep visual-inertial odometry with online error correction for rgb-d imagery," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2019.

[56] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 15–22.

[57] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*. IEEE, 2007, pp. 225–234.

[58] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[59] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2320–2327.

[60] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[61] A. Heyden and M. Pollefeys, "Multiple view geometry," *Emerging topics in computer vision*, pp. 45–107, 2005.

[62] M. I. Lourakis and A. A. Argyros, "Sba: A software package for generic sparse bundle adjustment," *ACM Transactions on Mathematical Software (TOMS)*, vol. 36, no. 1, p. 2, 2009.

[63] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski, "Bundle adjustment in the large," in *European conference on computer vision*. Springer, 2010, pp. 29–42.

[64] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz, "Multicore bundle adjustment," in *CVPR 2011*, June 2011, pp. 3057–3064.

[65] H. Liu, M. Chen, G. Zhang, H. Bao, and Y. Bao, "Ice-ba: Incremental, consistent and efficient bundle adjustment for visual-inertial slam," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1974–1982.

[66] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.

[67] K. Lindgren, S. Leung, W. D. Nothwang, and E. J. Shamwell, "Boom-vio: Bootstrapped monocular visual-inertial odometry with absolute trajectory estimation through unsupervised deep learning," in *2019 19th International Conference on Advanced Robotics (ICAR)*. IEEE, 2019, pp. 516–522.

[68] R. Szeliski, "Prediction error as a quality metric for motion and stereo," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2. IEEE, 1999, pp. 781–788.

[69] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," 2013.

[70] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, "On measuring the accuracy of slam algorithms," *Autonomous Robots*, vol. 27, no. 4, p. 387, 2009.

[71] P. Sinha and R. Russell, "A perceptually based comparison of image similarity metrics," *Perception*, vol. 40, no. 11, pp. 1269–1281, 2011.

[72] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[73] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018.

Appendix A

MODEL-BASED VS. MODEL-FREE EXAMPLE

IMU Integration

Mappings from IMU gyroscope and linear accelerometer measurements to orientation and position are one such area with known models. Integrating gyroscope measurements for estimating orientation changes in the body frame works particularly well, as is shown in Figure A.1 with a sample trajectory from the popular KITTI odometry dataset [51].

Changes in position from linear acceleration measurements, however, are considerably less reliable. Figure A.2, collected from the same KITTI trajectory, display the levels of drift as well as difficulty in removing the gravity vector experienced when directly integrating accelerometers. convolutional neural networks (CNNs) prove useful, though, as an effective strategy for recovering initial velocity, since this is unobservable from integrating accelerometer readings (only provides change in relative velocity).

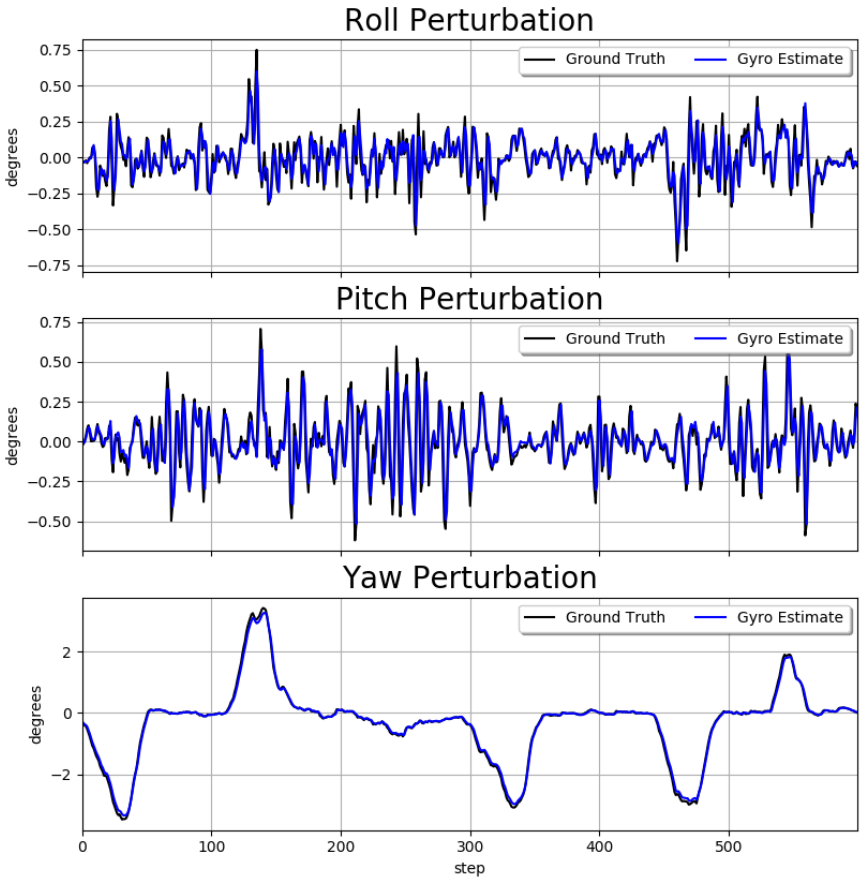


Figure A.1: Direct integration of gyroscope (angular velocity) measurements leads to accurate estimates of vehicle orientation perturbations in the body frame.

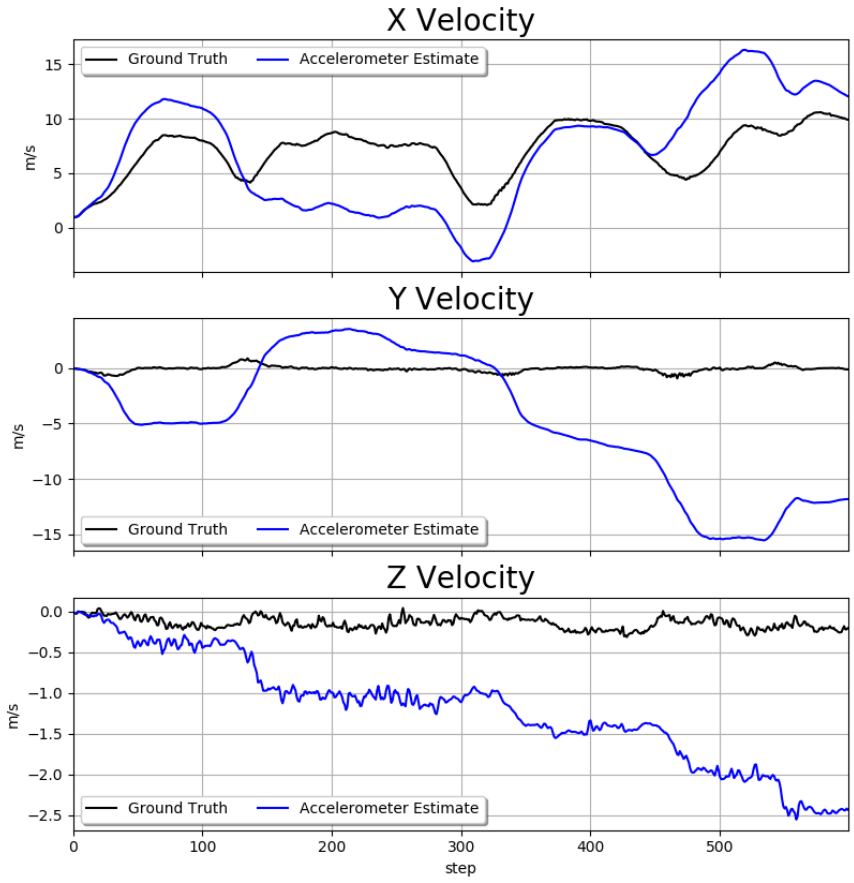


Figure A.2: Direct integration of accelerometer (linear acceleration) measurements leads to estimates of vehicle velocity with substantial error, even when starting with the ground truth initial velocity.