A Fused Convolutional and Recurrent Neural Network for Visual-Inertial Odometry

Justin Gorgen
A53096607
jgorgen@ucsd.edu

Haifeng Huang
A53208823
hah086@ucsd.edu

Hao-en Sung
A53204772
h3sung@ucsd.edu

Yen-ting Chen
A53214417
jgorgen@ucsd.edu

Abstract

The results of a recent paper [1] on a neural network architecture for Visual-Inertial Odometry (VIO) are replicated to verify the performance of the algorithm. The algorithm achieves an error rate of 1% per on the KITTI 500m dataset, and is able to tolerate offset in rotation, translation, and time between the IMU and camera.

1. Introduction

The problem of navigation is ancient. Both animals and humans have tackled this problem for milennia. Animals have evolved novel odometry solutions such as the desert ant *Cataglyphis fortis* using step-counting to find its way to and from its nest [10]. Meanwhile, ancient humans have developed absolute positioning methodologies such as celestial navigation or trade-wind-based localization [12] to navigate between Pacific Islands thousands of miles aparn. There is a distinction between the kind of navigation the ant performs when counting steps, and finding the correct island in the middle of the pacific. The ant is calculating its relative position with respect to home, knowing only direction and distance, while the sailing navigator's goal is to find his absolute location on Earth. There are two types of navigation that can be calculated from the environment, absolute positioning or relative positioning. This paper will address the problem of relative positioning, or odometry, similar to the desert ant's step counting method or the odometer on a ca. Instead of counting steps, however, this paper will use an Inertial Measurement Unit (IMU) and a monocular camera system to calculate change in position over time.

2. A Background on Odometry

The most concise definition of odometry comes from [3], "Odometry is the use of data from motion sensors to estimate change in position over time." For ground vehicles, the most common method of calculating odometry is a wheel-encoder, which calculates turns of a vehicles wheels to estimate distance traveled. For air and sea vehicles, wheel encoders cannot be used, and instead these types of vehicles often use a combination of IMUs and pitot tubes that measure the local speed relative to the medium. An IMU is an electromechanical device that can measure rotations and accelerations in three-dimensions.

Visual-Inertial Odometry is the process of calculating odometry using an IMU and a camera. While not strictly a motion sensor, a single camera can be used to estimate change in position using aspects from multiple view geometry [6], a process referred to as "structure-from-motion." Both IMUs and cameras alone have several weaknesses as odometry sensors, but as demonstrated by more than a decade of research, fused measurements from IMUs and cameras can provide estimates of odometry more precise than wheel encoders or pitot-tubes.

2.1. Inertial Measurement Units and Their Shortcomings

Inertial Measurement Units come in many shapes and sizes, with sensing technologies ranging from spinning-wheel gyroscopes to million-dollar arrays of ring-laser gyroscopes to hundred-dollar micro-electromechanical systems (MEMS) devices[5]. The basic principle of using IMUs for odometry is to carefully sum changes in attitude and velocity and attitude over time to estimate the current velocity and attitude, and then integrate the current velocity to estimate the current change in position. A common method of calculating IMU odometry, from Paul Groves's *Principles of GNSS, Inertial and Multisensor Integrated Navigation Systems* [5], is to use a fifteen (15) state Kalman filter, with 3 states each for position, velocity, attitude, accelerometer bias errors, and gyroscope bias errors.

The main problems with using IMUs for odometry are the non-zero-mean errors of real accelerometers and gyroscopes, and that the integration algorithms require precise world models and motion models of the vehicle the IMU is on. Both of these problems stem from the fundamental nature of dead-reckoning with accelerometers, i.e. that small errors in acceleration turn into larger errors in veloc-

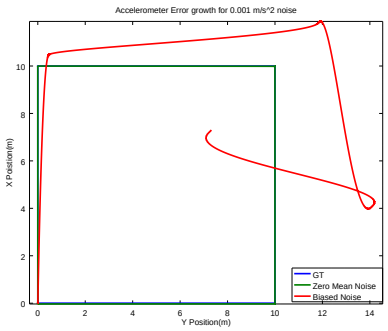

Figure 1: A simple simulation showing the effect of 0.001 m/s^2 Gaussian errors on acceleration for a 120-second long simulation. Zero mean errors have little effect on the overall shape of the trajectory, but biased accelerometer errors even as small as 0.001 m/s^2 grow quickly.

ity, which grow larger still when integrated into position.

The problem with integrating IMUs alone for odometry is that real accelerometers and gyroscopes have time-varying, non-zero-mean errors, called biases. Small bias errors in accelerometer and gyroscope measurements are integrated twice to calculate change in position, and thus small constant errors in acceleration become quadratic errors in position, as shown in 1.

An example of model difficulties with IMU involves handling gravitational anomalies. Surface gravity on earth varies by as much as $0.001ms^{-2}$ from the nominal $9.80ms^{-2}$. It is important to note that these gravity anomalies are not restricted to the vertical axis, e.g. large mountain ranges create gravity deflections in the horizontal direction. Therefore an IMU that does not properly compensate for gravity models, next to a large mountain range with a $0.001ms^{-2}$ gravity deflection would have a position error of 6.4 km after one hour, and a horizontal velocity error of 3.6 m/s. Compensating for gravity anomalies, either through precise models or additional sensors, is therefore important for use in IMU odometry.

2.2. Visual Odometry and Its Shortcomings

Visual odometry is the art of calculating changes in position using only optical sensors, typically using the cam-

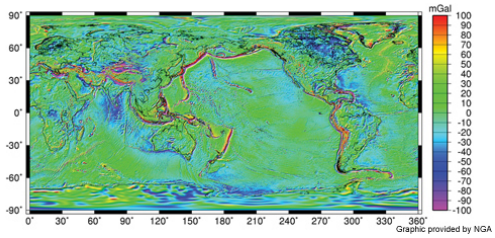

Figure 2: Gravity anomaly data from the National Geospatial-Intelligence Agency's EGM2008 survey. From http://earth-info.nga.mil/GandG/wgs84/gravitymod/egm2008/anomalies_dov.html Public Domain

eras in the infrared to ultra-violet range. While the celestial navigation used by ancient mariners is a form a visual odometry, modern interpretations involve using monocular or stereo vision sensors to estimate the motion of the camera or the vehicle to which the camera is attached. This is distinct from the tracking problem, where an moving object's motion is estimated from a static, off-platform camera. Research into monocular VO is limited, due to the inherent ambiguity of scale with a single camera. Meanwhile, stereo VO is much more mature, and NASA/JPL use stereo VO as a "critical vehicle safety system" on the Spirit and Opportunity Mars rovers [9]. Although critical safety system, during their missions Spirit and Opportunity did not use VO as a primary navigation aid, using it for only 14% of their trajectory. Due to issues with inclement weather and camera-recalibration, and the high reliability of other sensors, VO was mainly used to detect wheel slip.

Visual Odometry is very closely related to the computer-vision problem called structure from motion (SFM) and the control/estimation problem called Simultaneous Localization and Mapping (SLAM). However, both SFM and SLAM create and update maps of the environment, while VO has no such requirement of building a map of the environment. Thus, some approaches to solve the VO problem can be solved by a good SLAM or SFM algorithm, but at a higher computational expense. Recent research into using deep neural networks to solve the VO problem are an attempt to create a real-time, causal VO solution that does not have the extra burden of creating an explicit map of the environment.

2.3. Visual Odometry with Neural Networks

A few papers have examined the use of deep neural networks for Visual Odometry. A 2015 study by Konda and Memisevic [8] used the KITTI dataset to test a stereo visual odometry method. The study found that constructing a network to calculate rotations required novel gating activation functions for neural networks, but the network had significant position errors due to the accumulation of delta-

pose errors. This was because the network estimated change in position between each set of pictures, and accumulated these changes in position to calculate the final position. An attempt by [11] to perform monocular Visual Odometry with neural networks showed some success. This network learned to calculate the accumulated 6-DOF pose as a function of only a sequence of monocular images. However, it was unable to estimate pose for areas that did not visually overlap with the training, indicating over-training.

2.4. Visual-Inertial Odometry

Visual-inertial odometry is traditionally accomplished using a sensor fusion algorithm such as an Extended Kalman Filter (EKF) [7]. The best algorithms, such as the 2009 PhD dissertation from Eagle Sunrise Jones, perform on the order of 0.2% error per distance traveled, even using only a monocular camera. The traditional approach to visual-inertial odometry involves:

- Identifying salient features in each frame of a video sequence using a hand-tuned algorithm such as SIFT
- Matching features between each frame (ostensibly requires N^2 comparisons for matching N features, but various heuristics have been used to greatly speed this process)
- Storing and managing a database of features
- Calculating a camera pose from the visible features
- Calculating a correction for the feature locations to improve future position estimates

Each of these processes require hand-tuning based on the expected motion profile of the camera system. Additionally, the best-performing algorithms such as Jones [7], require a calibration maneuver in order to observe and calibrate the external calibration parameters of the camera i.e. the lever arm and rotation between the IMU and the camera. Nevertheless, the performance of the hand-tuned approach is very good, the only downside is the requirement of laptop-level CPU and GPU hardware to produce a real-time measurement.

A major breakthrough in machine-learning based VIO was published in January of 2017 [2]. This recurrent-neural-network based approach presented errors of 1% on the KITTI dataset, which is on the same order of magnitude of error performance as hand-tuned VIO algorithms. The approach in [2] can calibrate for offsets in time and space between the camera and IMU datastreams, and this approach is the method that will be replicated in this paper.

3. Dataset

In this section, we will inroduce the two dataset, KITTI and MAV dataset, that will be used to train the model in our experiment. We will go through the KITTI dataset in 3.1 and look into the MAV dataset in 3.2.

3.1. KITTI Vision benchmark

The odometry dataset provided by KITTI consists of 22 sequences, of which the first 11 sequences are labeled with ground truth. Each sequence contains frames ranging from 81 to 2000 frames. For every frame, a 3x4 ground truth transformation matrix which describes the the change between the coordinates of current frame and 0^{th} frame is provided. In addition to frames, the information of IMU is available for every frame, but only in several sequences. Since our modeled is designed for data with IMU information, we will discard the sequences without IMU paramteters in our experiment.

3.2. The EuRoC MAV Dataset

In MAV dataset, 11 sequences of frames which are captured inside two building. Each frame is also labeled with ground truth transformation matrix, and IMU data of each video is well measured. However, IMU-measuring and frame-capturing are asynchornized, while the frequency of measuring IMU parameters is about ten times higher than the one of capturing frames. In our experiment, we will utilize both the timestamps of frames and IMU data EuRoc provides to interpolate the IMU value of each frame.

4. Methodology

4.1. Network Framework

The network the authors used to implement sequence-to-sequence learning approach to visual-inertial odometry, VINet, is shown in Figure 3. The input to the network is monocular RGB images and IMU data which is a 6 dimensional vector containing the x, y, z components of acceleration and angular velocity measured using a gyroscope. The output of the network is a 7 dimensional vector - a 3 dimensional translation and 4 dimensional orientation quaternion - representing the change in pose of the robot from the start of the sequence.

In time step t, they fed in images in time step t and t+1 into two parallel convolutional neural network at one end of the network and concatenate the feature maps after three convolutional layers and apply convolutional layers to the concatenated feature maps, at last flatten it to get a feature vector of visual information. At the other end of the network, the authors fed in 6 dimensional vector of IMU data into IMU LSTM network and concatenated the output inertial feature vector of the IMU LSTM network with visual vector generated from convolutional neural network to form the core LSTM. The core LSTM not only took visual and inertial vectors and the state of the core LSTM of the last time step as input, but also was fed in the output of the core

LSTM of the last time step, which is the prediction of the translation and orientation quaternion. In the case of odometry estimation the availability of the previous output state is particularly important as the output is essentially an accumulation of incremental displacements at each step.

4.2. SE(3) Concatenation of Transformations

The pose of a camera relative to an initial starting point is conventionally represented as an element of the special Euclidean group SE(3) of transformations.

$$T = \left\{ \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix} | R \in SO(3), T \in R^3 \right\} \quad (1)$$

However, the Lie Algebra se(3) of SE(3), representing the instantaneous transformation,

$$\frac{\xi}{dt} = \left\{ \begin{pmatrix} [w]_\times & v \\ 0 & 0 \end{pmatrix} | w \in so(3), v \in R^3 \right\} \quad (2)$$

can be described by components which are not subject to orthogonality constraints. Conversion between se(3) and SE(3) is then easily accomplished using the exponential map

$$exp : se(3) \rightarrow SE(3) \quad (3)$$

The CNN-RNN thus performs the mapping from the input data to the lie algebra se(3) of the current time step. An exponential map is used to convert these to the special euclidean group SE(3) and sum it to the cumulated SE(3) of the last time step to get the cumulated sum of current time step. Figure 4 is the illustration of the SE(3) composition layer - a parameter-free layer which concatenates transformations between frames on SE(3).

4.3. Multi-rate LSTM

Because the IMU data(100 Hz) arrives 10 times faster than the visual data(10 Hz), the authors processed the IMU data using a small LSTM at the IMU rate, and fed that result to the core LSTM.

4.4. Optical Flow Weight Initialization

The authors initialized the convolutional neural network using imagenet at first, but they found that the convergence was slow and the performance is not very well, so they used the flownet which is trained to predict optical flow from RGB images to initialize the network up to the Conv6 layer and removed the layers which produce the high-resolution optical flow output and fed in only a $1024 \times 6 \times 20$ vector which they flattened and concatenated with the feature vector produced by the IMU-LSTM before being fed to the Core LSTM.

5. Experiment

In this section, we first introduce how we will pretrain our model in 5.1. Later, two different objective functions for SE(3) and se(3) respectively are shown in 5.2. After that, experiment procedures and expected results for both MAV and KITTI dataset are listed in 5.3 and **??**.

5.1. Pretrained Model

According to original paper, they tried to use pretrained ImageNet model as their based model but experienced slow model convergences and poor test performance. In view of this, they pretrained their model with optical flow dataset from RGB images [4].

5.2. Objective Function

Based on the authors, directly using SE(3) as teaching signal makes the model easily stuck on local minimum. To overcome this issue, they proposed to also optimize se(3). Two optimization formulas can be written as follows.

$$\mathcal{L}_{se(3)} = \alpha \sum \|\omega - \hat{\omega}\| + \beta \|v - \hat{v}\|,$$
$$\mathcal{L}_{SE(3)} = \alpha \sum \|\boldsymbol{q} - \hat{\boldsymbol{q}}\| + \beta \|T - \hat{T}\|,$$

where

ω is the parameterized instantaneous rotation matrix

v is the instantaneous translation matrix

$\boldsymbol{q}$ is the 4-dimensional orientation quaternion

T is the translation matrix

From experiments in original paper, iteratively updating the model with both optimization formulas helps model achieve better performance than using only either one of them. The expected result is shown as Fig. 5.

5.3. Datasets

Since two datasets are taken in different scenario, we will train the model for each dataset seperately. Both datasets contains 11 different sequences of frames and corresponding IMU data. We will introduce how we split the both datasets to fit our model use in following subsetions.

5.3.1 Dataset Split

Inspired by original paper, we can see the way of splitting data into training set and testing set has a huge impact on the resulting performance. There are two reasonable ways of dividing the dataset, one is taking several sequences as the training set, while leaving the others as the testing set. Another way is to divide each sequence into two parts, one for

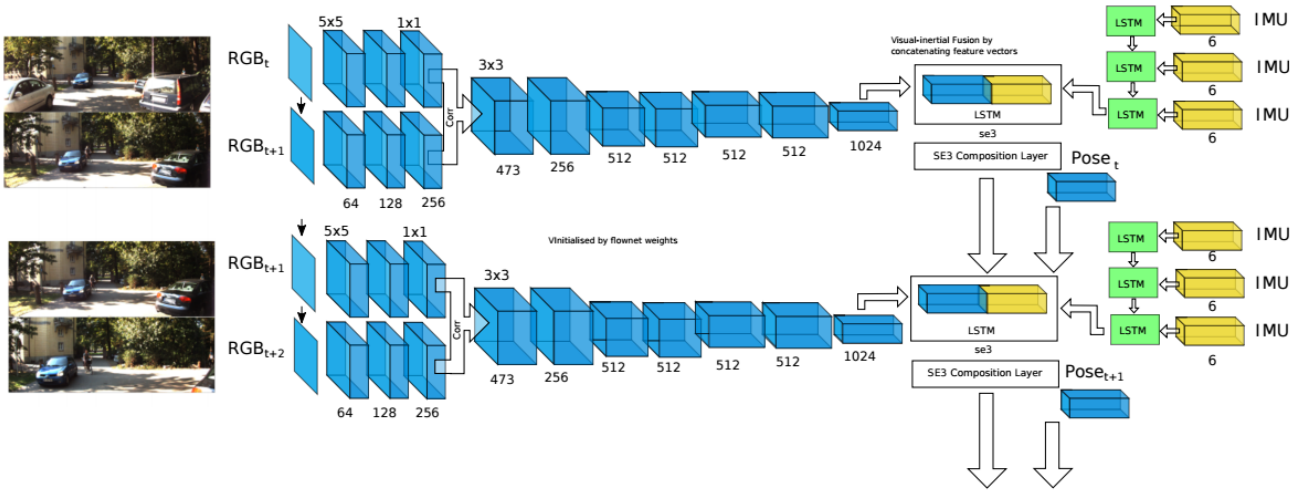

Figure 3: The proposed VINet architecture for visual-inertial odometry. The network consists of a core LSTM processing the pose output at camera-rate and an IMU LSTM processing data at the IMU rate.

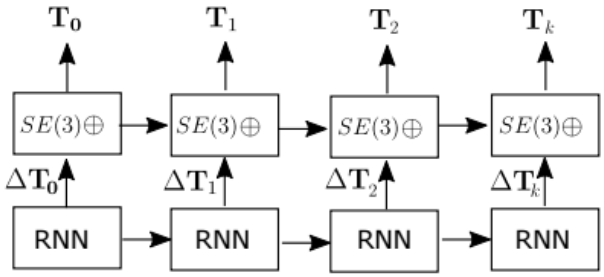

Figure 4: Illustration of the SE(3) composition layer - a parameter-free layer which concatenates transformations between frames on SE(3).

the training and the other for the testing. In our experiment, we will adopt both ways of splitting dataset into 80:20, and 50:50 portion, and then compare their results on both KITTI and MVA dataset.

5.3.2 Expected Results

From the original paper, the former stategy of dividing dataset is described as "Testing in known environment", while the latter one is described as "Testing in unknown environment". We can expect the result of the former one will be better than the latter one. However, we can examine our model by reviewing the result. The performance of the model trained under former strategy tells us if our design of model is feasible for visual odometry task, while the one trained under latter strategy implies how generalized our model could be.

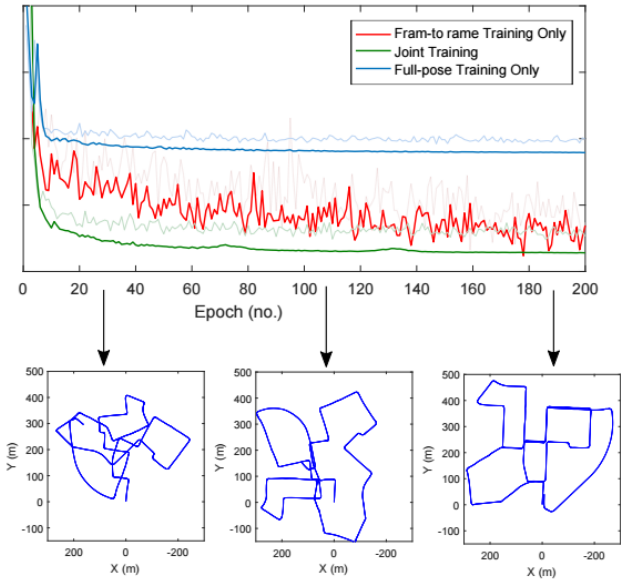

Figure 5: Expected Training Loss with Three Optimization Procedures: $\mathcal{L}_{se(3)}$ (left path), $\mathcal{L}_{SE(3)} + \mathcal{L}_{se(3)}$ (middle path), and $\mathcal{L}_{SE(3)}$ (right path)

6. Our Work

6.1. Schedule

6.2. Discussion

1. There are two ways to split the train-test dataset: split through time or split by instances. What is the performance of our model on this two different split strategy?

Table 1: Time Table

05/01	**Project Proposal Submission**
05/08	Implement main model with Keras 2.0
05/08	**Revised Project Proposal Submission**
05/15	Deploy model on AWS and run toy experiments
05/22	Finish experiments on KITTI and MAV datasets
05/22	Implement supplement codes, ex: visualization
05/29	Run experiments listed in discussion section
05/29	Visualize our results, ex: draw predicted paths
06/05	**Polished Project Report Submission**
06/12	**Final Project Report Submission**

2. There are two dataset: MAV and KITTI. Do they share any similar features?

3. In original paper, they work on only RGB dataset in coordination to pretrained optical flow model. However, RGB dataset is more costly in terms of processing and training. How is the model performance if it works on only gray-scale cameras?

4. Since camera has different frequency from IMU data, there are two ways to deal with this issue: down-sampling camera frames or up-sampling IMU data (duplicate). Which one will result in better performance?

5. Is there any better way to extend current neural network framework from using monocular camera to using stereo camera?

References

[1] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni. Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem. *CoRR*, abs/1701.08376, 2017. 1

[2] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni. Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem. *arXiv preprint arXiv:1701.08376*, 2017. 3

[3] C. Fairchild and T. Harman. *ROS Robotics By Example*. Packt Publishing, 2016. 1

[4] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*, 2015. 4

[5] P. D. Groves. Principles of gnss, inertial, and multisensor integrated navigation systems, 2008. 1

[6] R. Hartley and A. Zisserman. Multiple view geometry in computer vision, 2000. 1

[7] E. S. Jones. *Large scale visual navigation and community map building*. PhD thesis, University of California Los Angeles, 2009. 3

[8] K. R. Konda and R. Memisevic. Learning visual odometry with a convolutional network. 2

[9] M. Maimone, Y. Cheng, and L. Matthies. Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics*, 24(3):169–186, 2007. 2

[10] T. Merkle and R. Wehner. Desert ants use foraging distance to adapt the nest search to the uncertainty of the path integrator. *Behavioral Ecology*, 21(2):349, 2010. 1

[11] V. Mohanty, S. Agrawal, S. Datta, A. Ghosh, V. D. Sharma, and D. Chakravarty. Deepvo: A deep learning approach for monocular visual odometry. *arXiv preprint arXiv:1611.06069*, 2016. 3

[12] M. Walker. Navigating oceans and cultures: Polynesian and european navigation systems in the late eighteenth century. *Journal of the Royal Society of New Zealand*, 42(2):93–98, 2012. 1