Almost Visual Inertial Odometry

Francesco Olivato
Unimore
307009@studenti.unimore.it

Christine Blandinie Tsarre Signou
Unimore
258913@studenti.unimore.it

Nicola Quattrone
Unimore
307074@studenti.unimore.it

Abstract

The estimation task of visual odometry (VO) suffers from approximation errors. We address some of them by training a model to understand long term knowledge about physics by predicting sequence of inertial data. This approach forces the model to consider both the short term (frames) and long term (inertial sequence) knowledge during the motion estimation. We found out that our solution enhance the common VO models at inference time, without requiring any additional data in input.

Keywords
Visual Odometry (VO), Deep Learning

1. Introduction

Odometry is the process of estimating egomotion of an agent (e.g. vehicle, human or robot) in order to solve the position estimation problem in the environment [1]. The position estimation can be performed in different ways, for example by monitoring wheels movement, by measuring inertial data and also by analysing visual features.

We can therefore define different ways to perform the odometry task [1]:

- **Wheel odometry:** it is performed by monitoring the number of turns of wheels of an agent.
- **Visual odometry (VO):** it is made by incrementally estimating the pose of the vehicle through examination of the changes that motion induces on the images of its onboard camera.
- **Laser odometry:** it is similar to VO but it use consecutive laser scans.
- **Visual Inertial Odometry:** that is generally performed by coupling an IMU board and a camera to provide inertial data to the system.

There are other ways to perform the odometry task such as global positioning (GPS) but listing them is beyond the scope of this paper.

What is important to notice is that in some particular situation VO is very useful, for example in GPS-denied environments or in uneven terrain and other adverse conditions.

Altough, to make VO work effectively, there should be sufficient illumination in the environment and a static scene with enough texture to allow apparent motion to be extracted. Furthermore in classic VO the errors introduced by each new frame-to-frame motion accumulate over time. This generates a drift of the estimated trajectory from the real path that increase with the path length [1]. The VO drift can be reduced through combination with other sensor. This is the reason why VIO result to has better performance [2]. Onother way to enrich odometry is talk about Visual SLAM (simultaneous localization and mapping). This method is potentially much more precise, because it enforces more constraints on the path but not necessarily more robust [1]. Altough this method is much more complex and computationally expensive. Furthermore the VO philosophy is different and it cares about local conistency of the trajectory, whereas SLAM is concerned with global map consistency. As final consideration we can say that VIO is able to perform better in critical circumstances with respect to VO. Is a fact that inertial data enrich the position estimation during rapid movements, instead visual features are way more reliable during slow movements [2]. However is not always possible or suitable to introduce an IMU board inside a system.

VIO is now performed by classical and Deep methods starting from IMU. what is not know is the possibility to remove the IMU and build a monocular visual inertial odometry system predicting the inertial values from the same images used for Monocular VO.

We think that humans can overcome ambiguous visual stimuli by having a great perception of the world's physics, which allows them to fill in unclear information. The aim of this project is to reduce the approximation error in pure VO caused by sparse non-informative frames (e.g caused by motion blur, glare, ecc...) forcing the model to learn long term physics behaviours using lightweight sequences of inertial data.

2. Methods

We developed a deep learning solution which tries to mimic the previously described behaviour. Indeed we designed an inertial net that predict the current inertial values and an inertial buffer that stores longer term behaviour. Finally the net refines the odometry prediction with both inertial value and inertial buffer. We can see in figure 1 the block diagrams concerning the standard way to do VO [1] from which we based our pipeline described in the architecture section.

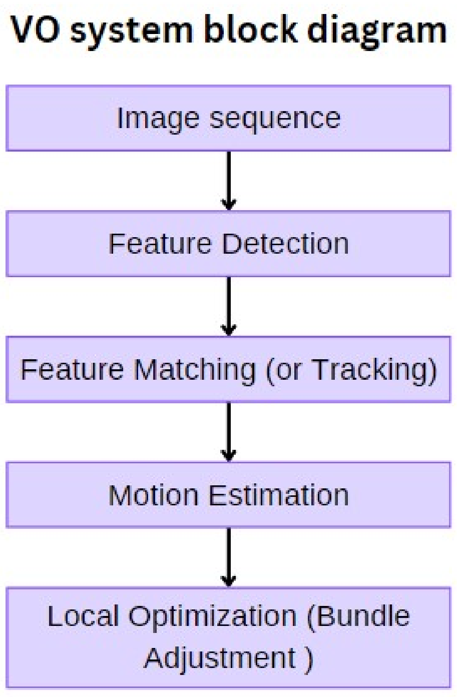

Figure 1. VO block Diagram main components

2.1. Dataset

It is used the ADVIO dataset because it provides 23 walking scenes with variable occlusion and a lot of sensory recordings. Among them it provides video, accellerometer, and ground truth for each timestep, that are necessary for our task. Moreover the data are captured from commodity but modern devices, which makes it a suitable dataset [1] for training a model for real world scenario.

2.2. Preprocessing

The frames were extracted from the video, they were resized from 1280x720 to 224x224 (to match model's input size) and normalized with min-max scaler.
Since the inertial data are sampled at 100Hz then the 60Hz video frames were resampled to 50Hz in order to have exactly 2 inertial samples for each frame.
Subsequently each timestamp sampled at 50Hz were paired to the frame of the closest timestamp of the 60Hz sampling.
It was built a 2 sec buffer of inertial data.
Finally timestamp of the accelerator, inertial buffer, video frames and ground truth were all syncronized.

A challenging aspect of the data preprocessing was to identify the errors in the accellerometer's data. Indeed the y-axis and z-axis of the accellerometer required rescaling from m/s^2 to g's in order to match the unit measure of the x-axis. [2]

2.3. Architecture

Our architecture 2 is composed by four different blocks:

- **Conv2d Odometry:** process a set of frames to extract odometry features.
- **Conv2d Inertial:** process a set of frames to extract inertial features.
- **Conv1d Inertial:** refine the inertial features (Conv2d Inertial) with respect to an inertial buffer.
- **Dense Odometry:** refines odometry features with respect to inertial values.

The following sections will describe the tecnical implementation of each block.

2.3.1 Conv2d Inertial and Odometry

Both Inertial and Odometry Conv2d feature exctraction are MobileNetV3. We decided to adopt this architecture because we are tackling an odometry task, which in real world scenarios often requires real-time capabilities on commodity devices. Indeed the model's contained size and fast inference makes it desirable in robotics application where odometry is essential.

[1]The dataset can be found at *https://github.com/AaltoVision/ADVIO*
[2]The data preparation's code was made public at the following repository *https://github.com/fmolivato/almost-vio*

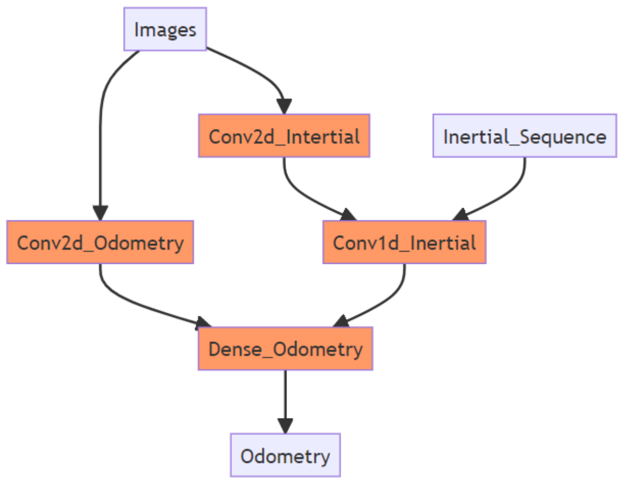

Figure 2. Our pipeline

2.3.2 Conv1d Inertial

The Conv1d Inertial block takes as input the Conv2d Inertial net's output, which provides the current inertial values and an inertial buffer of 100 element. This buffer stores a sequence of values that represent the inertial history. This block correlate the inputs in order to produce a current inertial values refined with respect of it's past.

During the training phase this sequence comes directly from the dataset,instead in the testing phase it's self-generated by the model.

The net is composed by six sequential convolutions, the kernel size of each convolution is (3) and the channels size increase progressively with the spatial reduction. According to the literature this configuration works well on convolutional architectures.

Each convolutional layer is initialized with the kaiming initialization [4], designed to work well with ReLU like activations therefore we used the SiLU activation for it's great empirical results [3].

2.3.3 Dense Odometry

The Dense Odometry block takes as input the output of Conv2D odometry and the output of Conv1d Inertial. This block is resposible to reduce approximation error of odometry values by correlating it with the refined inertial values.

2.4. Training

To train the network the data were shuffled with a fixed seed in order to ensure repeatability and then split in three subsets: training split with 80% of the dataset, validation with 20% and test with the remaining 20%. The training was performed with batch size of 32 and 20 epochs early stopping.
Since we are performing a regression task we decided to use the RMSE loss as cost function. No hyperparameter tuning were performed on the pipeline.

3. Results

The baseline used as benchmark reference for our solution is a simple MobileNetV3, it takes in input the frames and outputs the odometry values.
After the training session we've found out that the loss values of the baseline 3 appears to be more noisy, expecially after the 50th epoch, with repect to the inertial ones 4.

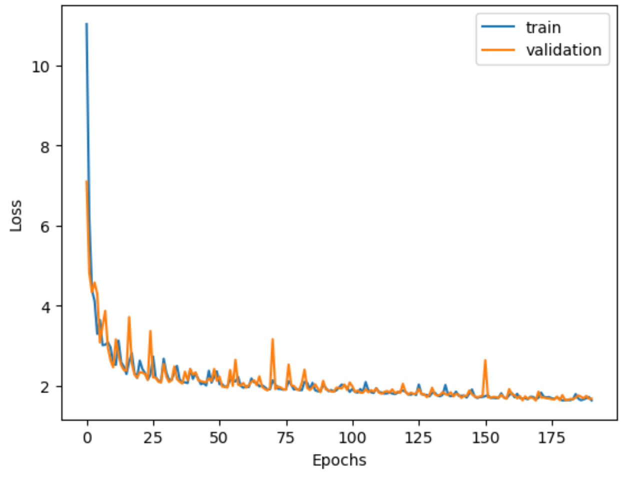

Figure 3. Baseline results

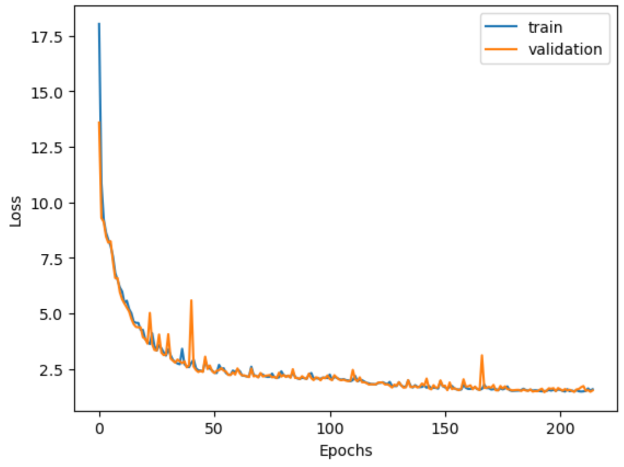

Figure 4. Our solution's results

Moreover we verified that out solution can successfully reduce the RMSE error with respect to the baseline. [3]

Solution	Stopping Epoch	Validation Loss (RMSE)
Baseline	191	1.64
InertialNet	215	**1.45**

4. Discussions

We confirmed that inertial knowledge is useful to solve successfully odometry tasks as suggested by the literature. Our contribution to the VO field is a method to extract inertial knowledge directly from visual data.

[3]The training's code was made public at the following repository *https://github.com/fmolivato/almost-vio*

References

[1] D. Scaramuzza and F. Fraundorfer *"Visual Odometry: Tutorial"*, IEEE Robotics Automation Magazine, vol. 18, no. 4, pp. 80–92, 2011. 1, 2

[2] D. Scaramuzza and Z. Zhang *"Visual-Inertial Odometry of Aerial Robots"*, Springer Encyclopedia of Robotics, 2019. 1

[3] Ramachandran, Prajit, Barret Zoph, and Quoc V. Le. "*Searching for activation functions.*" arXiv preprint arXiv:1710.05941 (2017). 3

[4] He, Kaiming, et al. "*Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.*" Proceedings of the IEEE international conference on computer vision. 2015. 3