RBE/CS 549 Computer Vision

P4 - Deep Virtual Inertial Odometry (Phase 2)

Uthiralakshmi Sivaraman
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester, MA, USA
usivaraman@wpi.edu

Noopur Koshta
Robotics Engineering Department
Worcester Polytechnic Institute
Worcester, MA, USA
nkoshta@wpi.edu

***Abstract*—The main aim of this project is to propose a method to solve stereo visual inertial odometry using deep learning based. The project is divided into three segments, Visual Odometry, Inertial Odometry and Visual Inertial Fusion. Various methodology and architecture are proposed to solve the problem. We use simple stacked layers of Convolution Neural Networks, Long short term network, Bidirectional Long Short Memory network, Temporal Convolution network to solve each of the division. The main objective is to propose a network architecture with custom loss function and data loading which performs visual inertial fusion seamlessly. Given the stereo data, IMU data, we aim to learn the camera pose through deep-learning.**

***Index Terms*—VIO, CNN, Bi-directional LSTM, Temporal Convolution**

I. Introduction

The objective of the project is to estimate the depth and scale of an environment using a combination of visual and inertial data. One solution is to use a stereo camera with a known pose, but this method is expensive and sensitive to motion blur. To overcome this, we can use an Inertial Measurement Unit (IMU) which measures linear and angular acceleration. A complementary approach to this is to use deep learning to fuse the visual and inertial data to estimate camera pose and backtrack depth.

Deep learning-based visual-inertial fusion is a powerful approach to estimate the camera pose and backtrack depth in real-time. It combines the strengths of both visual and inertial sensors to overcome the limitations of each individual sensor.

In visual-inertial fusion, the visual data provides high-resolution images that are used to estimate the camera pose and backtrack depth. The inertial data, on the other hand, provides measurements of linear and angular acceleration that can be used to estimate the camera motion and compensate for any motion blur that may be present in the visual data.

Deep learning algorithms are used to fuse the visual and inertial data in real-time. This involves training a neural network to predict the camera pose and backtrack depth using both visual and inertial data. The neural network takes the stereo images and IMU data as inputs and outputs the estimated camera pose and backtrack depth.

There are several challenges involved in training deep learning models for visual-inertial fusion. One challenge is that the visual and inertial data have different sampling rates and noise characteristics, which can make it difficult to fuse the data effectively. Another challenge is that the models need to be trained on a diverse dataset to ensure that they can generalize to new environments.

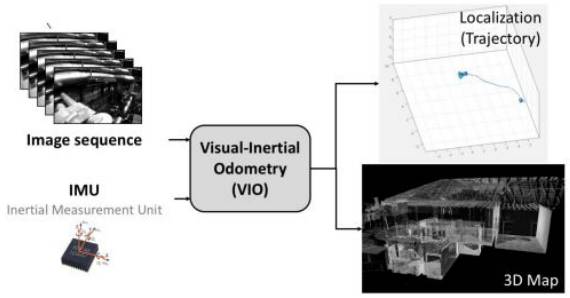

Fig. 1. Deep Learning based Visual Inertial Fusion

Despite these challenges, deep learning-based visual-inertial fusion has been successfully used in a variety of applications, including robotics, autonomous vehicles, and virtual reality. It has the potential to revolutionize the way we perceive and interact with the world around us, enabling new applications and capabilities that were previously impossible.

A. Literature Review

VINet is a deep learning-based method for visual-inertial odometry (VIO) that formulates the problem as a sequence-to-sequence learning task. The architecture includes a convolutional neural network (CNN) to extract visual features and a recurrent neural network (RNN) to process inertial measurements and the visual features. VINet minimizes a combination of geometric and photometric losses to learn accurate pose estimates and consistent visual features across different images. VINet achieves state-of-the-art performance on VIO benchmark datasets and can handle challenging environments where visual-only approaches may fail.

The End-to-End Learning Framework for IMU-Based 6-DOF Odometry estimates the position and orientation of a mobile device using only inertial measurements from its IMU. The framework includes data collection, network architecture

design, and training, with the network consisting of convolutional and recurrent layers to extract features from the IMU data and capture temporal dependencies. The network is trained using a loss function that penalizes differences between the predicted and ground-truth poses, with commonly used loss functions including Mean Squared Error and Geodesic Loss. Once trained, the network can be used for real-time pose estimation and has shown promising results in various settings, such as indoor navigation, augmented reality, and robotics.

DeepVO is an end-to-end deep learning approach for estimating visual odometry using a deep recurrent convolutional neural network (RCNN) that combines convolutional neural networks (CNNs) and recurrent neural networks (RNNs) to capture spatial and temporal information from image sequences. The CNN processes grayscale image sequences to extract features, which are then fed into a bidirectional long short-term memory (LSTM) network that predicts the camera's motion parameters. The model is trained end-to-end using a combination of translation loss, rotation loss, and regularization term. DeepVO outperforms traditional visual odometry methods and achieves state-of-the-art results on several benchmark datasets.

II. Background

Recurrent Neural Networks (RNNs) are a type of neural network that utilize not only the input data but also past hidden layers and output to keep track of previous inputs and corresponding decisions. However, RNNs struggle to learn and operate on long-term trends in input data using conventional training techniques, which limits their advantages over standard feed-forward networks. To overcome this, the Long Short-Term Memory (LSTM) architecture was introduced by adding gating cells that selectively store and forget memories. There are various versions of the LSTM architecture that have similar performance on real-world data. The memory cell contents are stored in ct, and the input gate controls the input's entry into the memory cell for the current time-step. The forget gate (ft) generates a control signal in the range 0 to 1 to clear the memory cell when necessary. The output gate (ot) determines whether the contents of the memory cell should be used at the current time-step.

Bidirectional LSTM (BiLSTM) is another variation of the LSTM architecture that has gained popularity in natural language processing (NLP) and speech recognition applications. Unlike traditional LSTM, which only processes the input data in one direction, BiLSTM processes the input data in both forward and backward directions. It involves two separate LSTM networks, one for processing the input sequence from left to right and the other for processing the input sequence from right to left. This allows the network to capture not only the past information but also the future information, making it better suited for tasks that require a more comprehensive understanding of the input sequence. BiLSTM has shown promising results in various NLP tasks such as named entity recognition, sentiment analysis, and machine translation.

Temporal convolution, also known as 1D convolution, is a technique used in deep learning for sequential data processing, similar to RNNs and LSTMs. In temporal convolution, a filter is applied to a sequence of input data, sliding along the sequence and computing a dot product at each time step. The resulting output is then passed through an activation function. The main advantage of temporal convolution is its parallel nature, allowing for efficient processing of sequences using modern hardware such as GPUs. Temporal convolution has shown promising results in various applications such as speech recognition, music processing, and action recognition in videos. It is also commonly used in conjunction with other neural network architectures such as CNNs and LSTMs to improve their performance in sequential data processing tasks.

III. Dataset

We are using Machine Hall Sequence dataset from EUROC MAV Dataset. Following are the key specifications of the data:

Visual-Inertial Sensor Unit

Stereo Images (Aptina MT9V034 global shutter, WVGA monochrome, 2×20 FPS)

MEMS IMU (ADIS16448, angular rate and acceleration, 200 Hz)

Shutter-centric temporal alignment

Ground-Truth

Vicon motion capture system (6D pose)

Leica MS50 laser tracker (3D position)

Leica MS50 3D structure scan

Calibration

Camera intrinsics

Camera-IMU extrinsics

Spatio-temporally aligned ground-truth

There are two types of datasets available for evaluation purposes. The first dataset includes ground truth data on 3D positions obtained from a Leica Multistation. It is designed to assess the performance of visual-inertial SLAM algorithms in a realistic industrial setting. The dataset was recorded in an unstructured and cluttered machine hall at ETH Zürich, making it particularly challenging to process.

The second dataset comprises ground truth data on 6D poses from a Vicon motion capture system and an accurate 3D point cloud of the environment, captured using a Leica 3D laser scanner. The environment includes obstacles to enhance the level of difficulty and provide more visual texture for reconstruction purposes.

IV. Methodology

A. Visual Odometry

Deep Learning (DL) has recently been dominating many computer vision tasks with promising results. Unfortunately, for the VO problem this has not arrived yet. In fact, there is very limited work on VO, even related to 3D geometry problems. We presume that this is because most of the existing DL architectures and pre-trained models are essentially designed to tackle recognition and classification problems, which drives deep Convolutional Neural Networks (CNNs) to extract

Fig. 2. Machine Hall Dataset

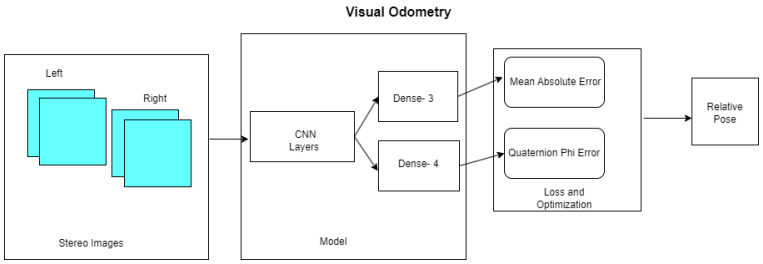

Fig. 3. Visual Odometry Methodology

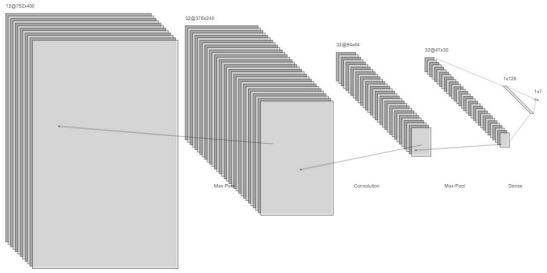

Fig. 4. Visual Odometry Architecture

high-level appearance information from images. Learning the appearance representation confines the VO to function only in trained environments and seriously hinders the popularisation of the VO to new scenarios. This is why the VO algorithms heavily rely on geometric features rather than appearance ones. Meanwhile, a VO algorithm ideally should model motion dynamics by examining the changes and connections on a sequence of images rather than processing a single image. This means we need sequential learning, which the CNNs are inadequate to.

DL has achieved promising results on some localisation related applications. The features of CNNs, for instance, have been utilised for appearance based place recognition[18]. Unfortunately, there is little work on VO or pose estimation. To our knowledge, firstly realises DL based VO through synchrony detection between image sequences and features.

After estimating depth from stereo images, the CNN predicts the discretised changes of direction and velocity by the softmax function. Although this work provides a feasible scheme for DL based stereo VO, it inherently formulates the VO as a classification problem rather than pose regression. Camera relocalisation using a single image is solved in by fine-tuning images of a specific scene with CNNs. It suggests to label these images by SfM, which is time-consuming and labour-intensive for large-scale scenarios. Because a trained CNN model serves as an appearance "map" of the scene, it needs to be re-trained or at least fine-tuned for a new environment. This seriously hampers the technique for widespread usage, which is also one of the biggest difficulties when applying DL for VO. To overcome this problem, the CNNs are provided with dense optical flow instead of RGB images for motion estimation. Three different architectures of CNNs are developed to learn appropriate features for VO, achieving robust VO even with blurred and under-exposured images. However, the proposed CNNs require pre-processed dense optical flow as input, which cannot benefit from the end-to-end learning and may be inappropriate to real-time applications.

1) Architecture: This neural network is specifically designed for image classification tasks and takes a RGB image of size 480x752 pixels as input. The architecture starts with an input layer and is followed by a convolutional layer with 32 filters and a kernel size of 3x3 to extract features from the image. The output is then passed through a max pooling layer with a pool size of 2x2 to avoid overfitting.

This process of applying convolution and pooling layers is repeated with another convolutional layer with 32 filters and a kernel size of 3x3 followed by another max pooling layer with a pool size of 2x2. The output is then flattened into a 1-dimensional vector and passed through a dense layer with 128 neurons to learn higher-level features that are useful for classification.

The output of the dense layer with 128 neurons is then passed through another dense layer with 7 neurons, the last two dense layers with 3 and 4 neurons are used to predict the relative position and orientation of objects in the input image.

B. Inertial Odometry

We propose a 6-DOF odometry method only with an IMU based on a neural network trained with end-to-end learning. The network architecture follows a convolutional neural network (CNN) combined with a two-layer stacked bidirectional long short-term memory (LSTM).

First, a 6-DOF relative pose expressed in the spherical coordinate system, or a 3D translation vector and an unit quaternion are used. Second, mean squared error (MSE), translation mean absolute error (MAE), quarternion phi error as pose distances are applied to the loss function in the network. Third, a multi-task learning that automatically balances the different metrics is integrated to handle the metrics for translation and rotation.

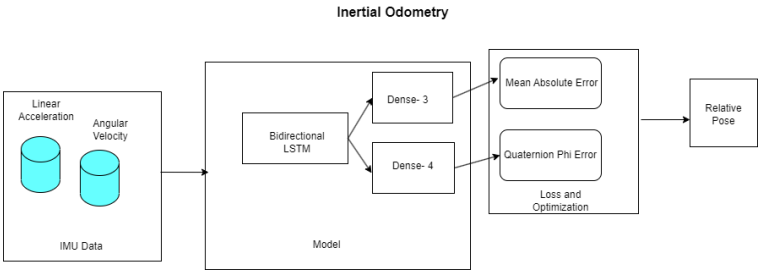

Fig. 5. Inertial Odometry Methodology

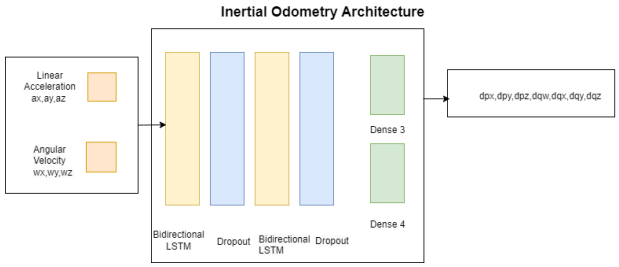

Fig. 6. Inertial Odometry Architecture

1) Architecture: This is a description of a neural network architecture that is ideal for sequence processing tasks, using a type of recurrent neural network called LSTM. The input to the network is a window of 200 frames of inertial data, containing 3-axis angular velocity and 3-axis acceleration. The input data is first processed separately by 1D convolutional layers with a kernel size of 11 and 128 features. Two convolutional layers are used, followed by a max pooling layer of size 3. The output of these layers is concatenated and passed to a bidirectional LSTM layer with 128 units, allowing both past and future IMU readings to influence the regressed relative pose.

A two-layer stacked LSTM model is then used, in which a bidirectional LSTM outputs a full sequence that is used as input for a second bidirectional LSTM. Dropout layers are added after each LSTM layer to prevent overfitting. Finally, a fully connected layer generates the output of a relative pose.

The architecture consists of several layers, starting with an input layer with dimensions of (None, 200, 6). This is followed by a bidirectional layer with 256 output neurons, a dropout layer, another bidirectional layer with 256 output neurons, and another dropout layer. The output is then passed through two dense layers, one with 3 output neurons and the other with 4 output neurons.

C. Loss Function

A straightforward approach to estimate the difference between ground truth and predicted poses is to compute their MSE. This is done when using the spherical coordinates representation, namely LMSE loss function. However, MSE is an algebraic rather than a geometric distance function.

1) Position Loss: The mean absolute error (MAE) is a common loss function used in regression problems to measure the average absolute difference between the predicted and true values. It is defined as the average of the absolute differences between the predicted and true values across all samples in the dataset. Mathematically, it can be represented as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} | y_i - \hat{y}_i |$$

Fig. 7. Mean Absolute Error

$$L = 1 - \left| \frac{1}{m}\sum_{i=1}^{m} \frac{\mathbf{y}_{\text{true},i}}{\|\mathbf{y}_{\text{true},i}\|} \cdot \frac{\mathbf{y}_{\text{pred},i}}{\|\mathbf{y}_{\text{pred},i}\|} \right|$$

Fig. 8. Cosine Dissimilarity Loss

where y_i and $\hat{y}_i$ are the true and predicted values for the i-th sample, respectively, and n is the total number of samples in the dataset. The MAE is a measure of the average magnitude of the errors made by the model, without considering the direction of the errors (i.e., whether the model is overestimating or underestimating the true values).

The MAE has several advantages as a loss function. It is relatively easy to interpret since it represents the average absolute difference between the predicted and true values. It is also less sensitive to outliers compared to other loss functions such as the mean squared error. However, it does not penalize large errors more heavily than small errors, which may be desirable in some applications.

2) Orientation Loss: Cosine dissimilarity loss is a common loss function used in machine learning tasks that involve learning similarity or distance metrics between pairs of data points. It is particularly useful for tasks such as image retrieval, face recognition, and recommendation systems.

The cosine dissimilarity loss is defined as the complement of the cosine similarity between two vectors, where the vectors are normalized to unit length. The cosine similarity between two vectors is a measure of their similarity, based on the angle between them. It is computed as the dot product between the two vectors, divided by the product of their magnitudes.

The cosine dissimilarity is simply the complement of the cosine similarity, defined as 1 minus the cosine similarity.

When used as a loss function, the cosine dissimilarity loss is typically computed as the average cosine dissimilarity between the predicted and true vectors over a batch of data points. The true vectors are typically one-hot encoded or label-encoded, and the predicted vectors are the outputs of a neural network or other machine learning model.

By minimizing the cosine dissimilarity loss, the model learns to map similar inputs to similar outputs, and dissimilar inputs to dissimilar outputs, effectively learning a similarity or distance metric between pairs of data points. This expression computes the dissimilarity (or distance) between the normalized y_true and y_pred vectors across all samples in the batch. The absolute value is taken to ensure that the dissimilarity is

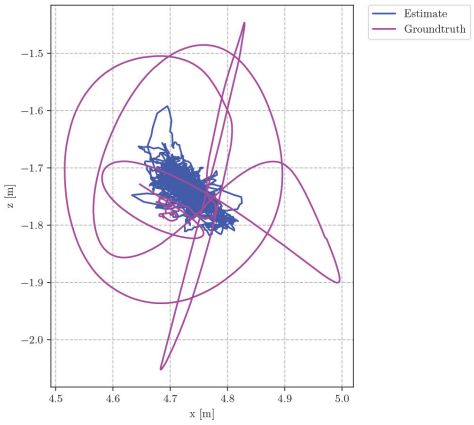

Fig. 9. Trajectory Plot - Side - Inertial Odometry

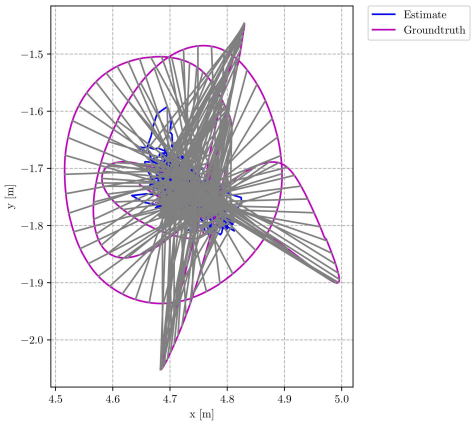

Fig. 10. Trajectory Plot - Top - Inertial Odometry

always non-negative. The value of the dissimilarity is then subtracted from 1 to obtain a similarity metric that ranges from 0 (completely dissimilar) to 1 (completely similar).

V. Results and Discussion

The following plots show the ground truth vs estimated trajectory plotted using rpg trajectory evaluation using Inertial Odometry and Visual Odometry.

A. Inertial Odometry

The plots show that there is a lot of difference between the estimated plots and ground truth plots in inertial odometry. There is a scale shift and time synchronization. The estimated trajectory is limited in a short range as compared to ground truth because of scale changes.

B. Visual Odometry

The plots show that there is a lot of difference between the estimated plots and ground truth plots in visual odometry. There is huge scale shift and time synchronization. The estimated trajectory is limited is negligibly visible or limitedly visible in this context as compared to ground truth because since we are plotting relative pose as comapred to absolute pose.

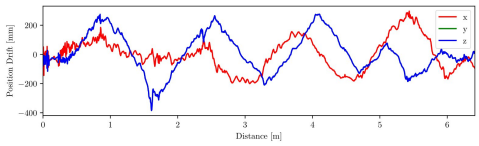

Fig. 11. Translation Error - Inertial Odometry

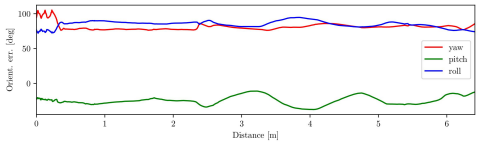

Fig. 12. Rotation Error - Inertial Odometry

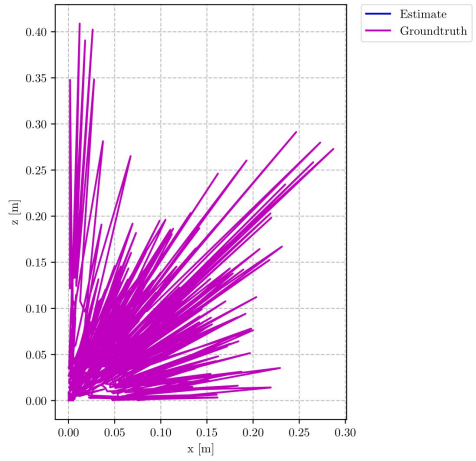

Fig. 13. Trajectory Plot - Side - Visual Odometry

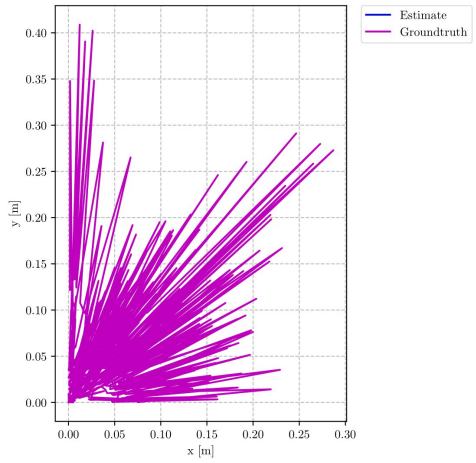

Fig. 14. Trajectory Plot - Top - Visual Odometry

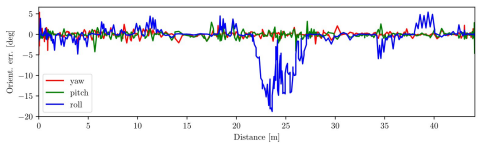

Fig. 15. Rotation Error - Visual Odometry

VI. Conclusion

In conclusion, our project aims to solve stereo visual inertial odometry through the use of deep learning. We have divided the project into three segments, namely Visual Odometry, Inertial Odometry, and Visual Inertial Fusion, and proposed various methodologies and architectures to address each segment. Our approach involves using simple stacked layers of Convolution Neural Networks, Long short term network, Bidirectional Long Short Memory network, and Temporal Convolution network. We aim to propose a network architecture with a custom loss function and data loading that seamlessly performs visual inertial fusion. Our objective is to learn the camera pose through deep learning using both stereo and IMU data. There are many challenges in the project. Primarily, there were issuing in time synchronization of IMU and Camera data and adding bias into IMU for bias correction. Also, since we used relative pose , we need to convert relative pose to absolute in order to synchronize with the ground truth.

VII. Future Work

The Recurrent Kalman Networks (RKN) is a powerful deep learning model that integrates the Kalman filter with a recurrent neural network to handle complex sequential data like speech and video. The model estimates the hidden state of the system using the Kalman filter and models the system's dynamics with the recurrent neural network. By utilizing factorized inference, RKN can estimate the covariance matrix of the system with fewer parameters. RKN has demonstrated promising results in several applications, including speech recognition and video analysis.

In Visual Inertial Odometry (VIO), RKN can be used to enhance the estimation process's accuracy and robustness. RKN can efficiently estimate the covariance matrix of the system in high-dimensional deep feature spaces by performing factorized inference, resulting in more accurate VIO. This can be particularly beneficial in challenging environments where traditional methods may struggle. Using RKN for VIO can improve the accuracy and reliability of localization systems in applications like robotics, autonomous driving, and augmented reality.

Extending Bidirectional LSTM to RKN combines the strengths of both models to create a more powerful deep learning architecture. Bidirectional LSTM can process data in both forward and backward directions, while RKN estimates the hidden state of the system and performs factorized inference in high-dimensional deep feature spaces. By integrating the Kalman filter into the forward and backward passes of the Bidirectional LSTM, the extended architecture can better model complex sequential data. This architecture has shown promising results in speech recognition and video analysis and can be useful in domains requiring the analysis of complex sequential data.

References

[1] https://rbe549.github.io/fall2022/proj/p4/

[2] Sun, Ke, et al. "Robust stereo visual inertial odometry for fast autonomous flight." IEEE Robotics and Automation Letters 3.2 (2018): 965-972.

[3] Mourikis, Anastasios I., and Stergios I. Roumeliotis. "A multi-state constraint Kalman filter for vision-aided inertial navigation." Proceedings 2007 IEEE international conference on robotics and automation. IEEE, 2007.

[4] https://github.com/KumarRobotics/msckf vio

[5] Silva do Monte Lima JP, Uchiyama H, Taniguchi RI. End-to-End Learning Framework for IMU-Based 6-DOF Odometry. Sensors (Basel). 2019 Aug 31;19(17):3777. doi: 10.3390/s19173777. PMID: 31480413; PMCID: PMC6749526.

[6] https://github.com/HTLife/VINet

[7] S. Wang, R. Clark, H. Wen and N. Trigoni, "DeepVO: Towards end-to-end visual odometry with deep Recurrent Convolutional Neural Networks," 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 2017, pp. 2043-2050, doi: 10.1109/ICRA.2017.7989236.

[8] Burri M, Nikolic J, Gohl P, et al. The EuRoC micro aerial vehicle datasets. The International Journal of Robotics Research. 2016;35(10):1157-1163. doi:10.1177/0278364915620033.

[9] Becker, Philipp, et al. "Recurrent kalman networks: Factorized inference in high-dimensional deep feature spaces." International conference on machine learning. PMLR, 2019.

[10] Zichao Zhang, Davide Scaramuzza: A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry, IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS), 2018.

[11] Kendall, Alex, Matthew Grimes, and Roberto Cipolla. "Posenet: A convolutional network for real-time 6-dof camera relocalization." Proceedings of the IEEE international conference on computer vision. 2015.