LAKEHEAD UNIVERSITY

MASTERS OF SCIENCE

Visual Simultaneous Localization and Mapping

A Research study submitted in fulfillment of the requirements for the degree of Masters of Science

in the

Department of Computer Science
Lakehead University,
Thunder Bay, Ontario, Canada

April 10, 2020

EXCEPTIONAL. UNCONVENTIONAL.

Authors

Edna Johnson
Gaurav Taluja
Japneet Sandhu
Ritika Chadha

Place for Project

Department of Computer Science
Lakehead University
955 Oliver Rd,
Thunder Bay, Ontario,
Canada, P7B 5E1

Supervisor

Dr. Thangarajah Akilan

Declaration of Co-Authorship / Previous Publication

1. Co-Authorship Declaration

I hereby declare that this dissertation incorporates material that is result of joint research, as follows: This dissertation also incorporates the outcome of a research under the supervision of Dr. Thangarajah Akilan. The key ideas, primary contributions experimental designs, data analysis, interpretation, and writing were performed by the author, and the contribution of the coauthors was primarily through the provision of proof reading and reviewing the research papers regarding the technical content.

We are aware of the Lakehead University Senate Policy on Authorship and we certify that we have properly acknowledged the contribution of other researchers to our dissertation, and have obtained written permission from each of the coauthors to include the above materials in our dissertation.

We certify that, with the above qualification, this dissertation and the research to which it refers, is the product of our own work.

2. Previous Publication

The dissertation includes two original papers that have been previously published or under review in peer reviewed journals and conferences, as follows:

Publication title/full citation	Publication status
Akilan T., Edna J., Gaurav T., Japneet S., Ritika C., ”Multimodality Weight and Score Fusion for SLAM”, IEEE CCECE, 2020	Accepted
Akilan T., Edna J., Gaurav T., Japneet S., Ritika C., ”A Hybrid Learner for Simultaneous Localization and Mapping”, IEEE SMC, 2020	Submitted

We certify that we have obtained a written permission from the copyright owners to include the above published materials in our dissertation. We certify that the above material describes work completed during our registration as graduate students at Lakehead University.

3. General

We declare that, to the best of our knowledge, our dissertation does not infringe upon anyone’s copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included our dissertation, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that we have included copy-righted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, we certify that we have obtained a written permission from the copyright owners to include such materials in our dissertation. We declare that this is a true copy of our dissertation, including any final revisions, as approved by our dissertation committee and the graduate studies office, and that this dissertation has not been submitted for a higher degree to any other university or institution.

Abstract

Simultaneous Localization and Mapping (SLAM) is used to simultaneously position and generate the corresponding map in the form of trajectory for the self-driving cars. SLAM has great applications in augmented reality and autonomous vehicles such as self-driving cars (SDC), navigation robots, and drones. The past decade has seen exciting and rapid progress in the field of solving SLAM problems by various implementation strategies. Hence, the developing enthusiasm for self-driving vehicles has given new headings to localization and mapping techniques. This research work focuses on the improvement of the performances by minimizing the errors of translation and rotation for an autonomous vehicle. The improvement is shown in three stages of implementation. The first stage incorporates unimodality-based learning with seven different unimodals (ResNet18, ResNet34, ResNet50, ResNet101, VGG16, VGG19, and AlexNet). The second stage is multimodality-based learning, as fusion techniques are applied respectively to the best two unimodals for early fusion and five unimodals for late fusion. In late fusion, the predicted scores are fused using average mathematical operation whereas, in early fusion, weight layer enhancement is performed using addition and multiplication mathematical operations. The third stage is a hybrid learner in which early fusion and late fusion are performed consecutively on ResNet101 and VGG19. This report consists of a comparative and experimental analysis of all the models and their relative translation and rotational error improvements. The results also comprise of the trajectories of the respective models in order to visualize the mapping. The experimental analysis demonstrates that the hybrid fusion model (M15) gives better results than unimodal, early, and late fusion models. Even though there are related works utilizing fusion, however, Apolloscape dataset has never been utilized to perform fusion strategies, which makes this work distinctive and insightful. The research aims to prove that fusion methods achieve better results than the unimodality-based learning. Henceforth, this project report provides a more reliable and effective DCNNs for SLAM.

Keywords

SLAM, deep learning, multimodal learning, hybrid learner

Acknowledgements

We would like to express our sincere gratitude to several individuals for supporting and assisting us throughout this journey of the Research Project. First and foremost, we offer our most profound gratitude to our supervisor, Dr. Thangarajah Akilan, who has been a constant support throughout our project with his patience, knowledge, insightful comments, invaluable suggestions, and creative ideas. His extensive understanding and professional expertise in Machine Learning techniques enabled us to extend our project making it worthwhile for the submissions to various conferences. We are grateful to him for his valuable time in directing us, providing the best possible solutions for infinite queries and furthermore for improving our presentation and writing skills.

We would like to express our genuine thanks to Lakehead University for providing us the safest and secure environment and also providing fundamental courses that helped us in building up the foundation blocks of our project. Our sincere praise for the Department of Computer Science, Lakehead University for their uniform support and timely feedback.

We would also like to thank our friends and family for their unwavering support and encouragement throughout the journey of these eight months. Finally, we thank god for his abundant blessings and for letting us through all the difficulties.

Acronyms

SLAM - Simultaneous Localization and Mapping
DCNN - Deep Convolutional Neural Network
SDC - Self-driving Car
SDV - Self-driving Vehicle
CNN - Convolutional Neural Network
DL - Deep Learning
AR - Augmented Reality
ANR - Autonomous Navigation Robots
GPS - Global Positioning System
DoF - Degree of Freedom
AF - Activation Function
NN - Neural Network
EKF - Extended Kalman Filter
LSTM - Long Short Term Memory
ILSVRC - ImageNet Large Scale Visual Recognition Competition
FC - Fully Connected
SVM - Support Vector Machine
MAE - Mean Absolute Error
MSC - Mean Squared Error
NRMSC - Normalised Root Mean Squared Error
RMASE - Relative Mean Absolute Scaled Error
MASE - Mean Absolute Scaled Error
MAPT - Mean Average Processing Time

Contents

Chapter 1

Introduction

SLAM is a technological process that enables a device to build a map of the environment and, at the same time, helps compute the relative location by using a map. The applications of SLAM range from self-driving cars to spatial exploration. The SLAM for Self-Driving Vehicles (SDV) incorporates the comprehension of the vehicle's location and its area to get the direction of its movement. As these location coordinates hold significant importance, henceforth, the slightest miscalculation may be lethal to the traveler or the environment of the independent vehicle. There are various techniques to do self-localization and mapping. However, the Deep Learning approaches have surmounted the others by their proficiency in the extraction of the finest features and providing better accuracy results.

This implementation focuses on the self-localization module of the Apolloscape Dataset. The dataset consists of the images and poses, additionally including the ground truth values to make a legitimate comparison. PostNet architecture is adopted to improve Framework accuracy, which is a Convolutional Neural Network (CNN) model that helps in the continuous posture estimation, making it a perfect counterpart for the execution. This research can be segregated into three tasks, where each task is an extension of the previous one. The first task concentrates on a comprehensive experimental study of the several Deep Learning (DL) methods for SLAM. The second part is dedicated to the ablation analysis of the proposed multimodal fusion approaches:

1. Early fusion via layer weight enhancement of the feature extractors
2. Late fusion via score refinement.

The third part of the research is committed to the proposed hybrid learner, which is the combination of early and late fusion techniques into a single model arrangement. The proposed early fusion model harnesses two pretrained DCNNs, viz., ResNet101, and VGG19 for performing weight enhancement, and the late fusion model uses five pretrained DCNNs, viz.., ResNet18, ResNet34, ResNet50, ResNet101, VGG16 and VGG19 as the feature extractors for the pose regressor. The hybrid learner combines the results of the early and late fusion on ResNet101 and VGG19.

1.1 Background

SLAM is a technique used to build a consistent map of the environment and determining the location at the same time [1]. It provides solutions for placing an object robotically controlled in an unknown location, thereby demonstrating its great potential in Augmented Reality (AR), autonomous vehicles. For example, self-driving cars, drones, unmanned air vehicles, and Autonomous Navigation Robots (ANR). Self-driving cars are automated vehicles that are remotely controlled by sensing the environment through various sensors such as lidar, radar, sonar, etc.

The most fundamental task with respect to self-driving cars is determining the location at all times [2]. Because, the slight ambiguities during the calculation of the location coordinates will have catastrophic results in comparison to the human-crewed vehicles, as autonomous vehicles do not possess unbiased and anti-calamity decision making power as the human-crewed vehicles. It is not only crucial for self-localization but also for path-planning, scene-understanding, and navigation problems as well.

Various research work has been carried out in the SLAM field. But, the most commonly known technology for self-localization is Global Positioning System (GPS), which relies on the satellites to provide the location information coordinates. Even though GPS is quite cheaper, yet it has limitations in restricted spaces such as near skyscrapers, underwater, indoors, tunnels, or places having jammed signals, which makes it insufficient to support SLAM applications. Moreover, community-driven maps such as OpenStreetMaps (OSM) [2] are also employed in localizing the autonomous vehicles and tracking the position. But, even it is not potentially adept as it has tremendous overhead in computation, communication, and memory requirements.

1.2 Problem

Autonomous vehicles are one of the significant technological developments in the history of invention. Although experiments have been performed on this technology since the 1920s, but the promising trials have taken place since the 1950s, and there have been steady improvements in this field since. After being in the spotlight for so many years, yet, the acceptance rate of autonomous cars is less as the dysfunctionality of these machines can lead to unfortunate consequences. Thus, the problem statement arises from this query, focusing on the measures to improve the precision of the vehicle in terms of its location and to maximize its accuracy. The next challenge associated is an extensive understanding of the underlying architecture as to which technology can provide better results with the dataset chosen. Handling the vast dataset requires an efficient execution procedure. An efficient framework is needed to process the dataset that comprises of numerous images to adhere to the high computational speed. The inference time taken during the running of the framework needs to be measured to compare it with the time taken for better accuracies. Concerning the size of the dataset, storage also needs to be considered.

1.3 Purpose

Self-Driving Vehicles (SDVs) are autonomous machines that can perceive the surroundings and navigate themselves without human intervention. As this application is an ongoing issue, therefore, there is an increasing need to achieve better accuracies. Reduced accuracy is fatal to humans as well as to the environment because of the consequences of the damage that can be caused. There are different methods embraced on account of SDVs. However, each one of them doesn’t meet the expected outcomes. So, the motivation behind this is to accomplish better accuracies utilizing different profound learning strategies in order to find the location coordinates of the vehicle, which includes the estimation of translation and rotation errors.

1.4 Goal

This research focuses on the improvement of the self-localization module of the Apolloscape dataset by utilizing the PoseNet framework. The foundation of the research work is a comparative analysis of the seven pretrained models using PoseNet regressor. The proposed model for improving accuracy is discussed under multimodality based learner viz., early and late fusion, and hybrid learner. The deliverables from this project are translation and rotation errors along with the trajectories. The translation and rotation errors are calculated in terms of mean and median, while the timing analysis is performed by calculating the mean average per sample processing time.

1.5 Dataset

Various datasets are targeted for visual tasks used for robot navigation (visual indoor SLAM), which includes localization, 3D geometry estimation, lane segmentation, and detection. Few of the famous datasets for the visual indoor SLAM are UTIAS Multi-Robot cooperative localization and Mapping dataset, and Intel research lab. However, concentrating on the self-driving technique, a set of comprehensive visual tasks are favored to be collected consistently within a consolidated dataset.

For the past few years, numerous datasets have been gathered in different spots, endeavoring to expand the variability and intricacy of urban road views for self-driving applications. Berkeley DeepDrive BDD100K is one of the most significant datasets to date and contains $100,000$ recordings of more than $1,100$-hour driving occasions across different times, and changing climate conditions. This dataset is taken in New York and San Francisco. Few other famous datasets include Comma.ai, Oxford's Robotic Car, and Cityscape.

The dataset used in this research is Baidu Apolloscape. It is one of the massive datasets taken in China, which defines 26 distinct semantics items such as bicycles, cars, pedestrians, etc. In terms of data scale apolloscape dataset is unique, the granularity of labeling, and task variations in real-time environments. There are four categories of the dataset available from Baidu's:

1. Scene Parsing
2. Car Instance
3. Lane Segmentation
4. Self- Localization

To collect the static 3D environment, they adopted Reigl VMX- 1HA as an acquisition system that consists of two VUX-1HA laser scanners, one VMX-CS6 camera system with resolution 3384×2710 and a measuring head with IMU/GNSS which includes position accuracy $20 - 50mm$, roll and pitch accuracy $0.005°$. The whole system is installed on the roof of a mid-size SUV and has been internally calibrated and synchronized. Additionally, the system contains two high frontal cameras capturing with a resolution of 3384×2710 and is well-calibrated with the LIDAR device [3].

As per our requirement, we worked on the self-Localization module of the Apolloscape dataset. The self-localization model consists of various submodules of which self-localization-examples is selected. It consists of four folders.

1. Images - The image folder consists of records 1 to 14 containing images from cameras 1 and 2 with a total number of 2999 images.
2. Pose - The folder contains the pose values for the images for each corresponding record and camera number. It defines 6-DOF (Degree of freedom) values with are $roll$, $pitch$, yaw, x, y, and z, which indeed define the translation and rotation parameters.
3. Camera params - It defines the essential parameters of the camera used for capturing the data.
4. Train-val split - It contains the training and validation split data used for the experimental analysis.

Chapter 2

Literature Review

2.1 Convolutional Neural Network (CNN)

Artificial Intelligence and Machine Learning were actively researched, especially in these modern times, to reduce the gap between the capabilities of machines and humans. There has been tremendous progress in algorithms like Convolutional Neural Network (CNN) in the domain of computer vision.

CNN is an artificial neural network that imitates the brain of the human brain, as neurons are organized like an arrangement of neurons in the human brain. It learns the hierarchies of features through backpropagation by using a set of layers such as convolution layers, dropout layers, average pooling layers, etc. It takes an image as input, assigns weights, and biases to the different objects in the image, which are then learned in the training phase through backpropagation algorithms. Rikiya et al. in [4], discuss basic concepts of CNN in various domains like radiology. This paper discusses the fundamental concepts of CNN applied to radiology, challenges encountered, and future directions. Authors talk about two major concerns related to the application of CNN in radiological tasks, such as overfitting and small dataset. Some of the solutions proposed by them to mitigate the challenges are to increase the training data size, using adequate hyperparameters such as batch size, regularization with weight decay or dropout, and data augmentation.

The functionality of CNN is incomplete without Activation Functions (AFs) as it introduces non-linear properties in the network. Without activation functions, the model is just a simple linear regression model that is incapable of learning complex mappings during backpropagation, making the sole purpose of CNN model futile. AFs perform computations between the hidden layers and the output layers in the Deep Learning (DL) architecture [5]. It performs computations on the input, weights, and biases, and the output of the computation is then fed as input to the successive layer in the DL architecture. Though there are many types of activation functions, but the most famous ones are as follows – Sigmoid or Logistic, Tanh and RelU.

Chogozie et al. in [5], surveys the usage of various AFs and the recent trends in the usage of the AFs in DL. Authors give an extensive survey of the majority of the activation functions in their paper and compare them with the state-of-the-art results. The authors give a detailed explanation of Rectified Linear Unit (ReLU) Function in [5]. ReLU was proposed by Nair and Hinton in 2010, and since then, it has been one of the most popular AFs. It represents a linear function, and therefore, preserves the properties of the linear model, which makes them easy to optimize, with gradient-descent methods [6]. The RelU activation function performs a threshold operation by setting the values zero if they are less than zero. And thus, is given as below:

$$f(x) = max(0, x) = \begin{cases} x_i, & \text{if } x_i >= 0 \\ 0, & \text{if } x_i < 0 \end{cases} \tag{2.1}$$

It forces the values less than zero to be set as zero, thereby, reducing the problem of vanishing gradient problems as seen in other activation functions such as tanh function, sigmoid function, etc.

2.2 Transfer Learning

Transfer Learning is a concept in machine learning, based on the fact that learning from the older task is applied to a newer related problem. For example, if an individual knows how to ride a bike, then he can learn to drive a car also. This is one of the examples of transfer learning in a very general way. It is very different from traditional learning, as, in latter, the models or tasks are isolated and function separately. They do not retain any knowledge, whereas Transfer Learning learns from the older problem and leverages the new set of problems. Matthew et al. in [7] gives a detailed survey about various transfer learning methods in Reinforcement Learning (RL) paradigm. It surveys and groups transfer learning methods on the basis of the goal of the transfer method, metrics to measure the success, and what information should be transferred. It discusses various metrics to analyze the performance of the transfer learning method, such as jump-start, asymptotic performance, total reward, time to the threshold, and transfer ratio.

Wenyuan et al. in [8] present a novel transfer learning algorithm called TrAdaBoost, to extend a boosting-based learning algorithm. It is handy in the domains when a new problem from a new dataset arises, but there exists a labeled data from the similar old dataset. It comes with two issues, firstly, labeling the new data would be too costly, and no proper use of old data would lead to a waste of data. Now, this new algorithm proposed by authors in this paper, would take a small amount of new data and utilizing a big chunk of old data to construct a high-quality classification model. In this way, TrAdaBoost effectively transfers knowledge from old data to new data.

2.3 SLAM

Simultaneous Localization and Mapping is an active research domain in robotics and artificial intelligence. As discussed in the section background of chapter I, SLAM enables a remotely automated moving vehicle to be placed in an unknown environment at an unknown location. According to Whyte *et al.* [9] and Montemerlo *et al.* [10], SLAM is building a consistent map of this unknown environment and determining the location relative to the map. Through SLAM, robotics and semi-automated vehicles can be truly and completely automated without any or minimal human intervention. But the estimation of maps consists of various other entities such as large storage issues, precise location coordinates, which makes SLAM a challenging task, especially in the real-time domain.

Many researches have been done worldwide to determine the efficient method to perform SLAM. In [11], Montemerlo *et al.* proposes FastSLAM algorithm as an efficient solution to the SLAM problem. FastSLAM is a recursive algorithm that calculates the posterior distribution spanning over autonomous vehicle's pose and landmark locations, yet, it scales logarithmically with the total number of landmarks. This algorithm relies on an exact factorization of the posterior into a product of landmark distributions and a distribution over robot paths. However, Montemerlo *et al.* [10] describes a new and more refined method that overcame the drawbacks of the original FastSLAM method proposed in [11]. This newly proposed method makes a straightforward modification from the original FastSLAM method when proposing a new robot pose; it does not only take into consideration the motion estimates but also the most recent sensor measurements. In this way, they converge the original FastSLAM algorithm for linear SLAM problems. The latest research on SLAM originates from the paper by Smith and Cheeseman [12], proposing the use of the Extended Kalman Filter (EKF). It is based on the notion that pose errors and errors in the map are correlated, and the covariance matrix obtained by the EKF represents this covariance. Extensive research and experiments in robotics to make them completely automated resulted in the landmark paper by Smith, Randall, and Cheeseman [13].

This paper discussed that a moving autonomous vehicle builds its map through an unknown environment by taking relative observations of landmarks, wherein a common error correlates the estimated coordinates of these landmarks in the vehicle's location. There are two main approaches for the localization of an autonomous vehicle: metric SLAM and appearance-based SLAM [14]. However, this research focuses on an appearance-based SLAM that is trained by giving a set of visuals collected from discrete locations.

2.4 PoseNet

The neural network is a technology that comprises of a network that has weights associated with it. The weights are adjusted through a series of trials and experiments in the training phase so that the network can learn and can be used to predict the results for the real-time test data. There are various kinds of neural networks available such as Feedforward Neural Network, Radial Basis Neural Network, Multilayer Perceptron, Convolutional Neural Network, Recurrent Neural Network, etc. FeedForward Neural Network is the deep learning models which aim to approximate a function. They are called as feedforward because information flows from input through the intermediate computations to the final output. It does not involve any feedback connections in which the outputs of the successive layer are fed to the preceding layer [15].

But Martin *et al.* [16] proposes an illustration where Long Short Term Memory (LSTM) Neural Networks are more efficient than feedforward Neural Network in the domain of Language Modelling. It is because of the underlying fact that feedforward Neural

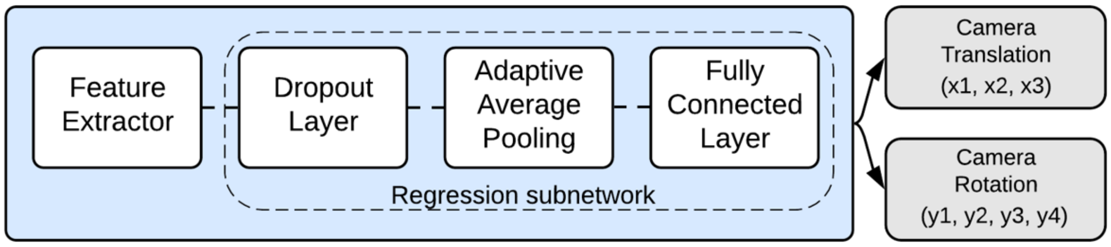

Figure 2.4.1: PoseNet Architecture

Network takes into account a fixed length to predict the next word. In contrast, LSTMs exploit the full length taking into consideration all the predecessor words. But, LSTMs are hard to train using backpropagation through time [17]. Another well-known issue associated with LSTM is vanishing gradient problem, which means that gradient fed back to the network through backpropagation either decays or grows exponentially [18].

But the PoseNet is a DCNN, which overcomes all the above-discussed issues by taking an image as the input and regressing the poses of the image taken. Furthermore, it is observed that PoseNet is a robust model that localizes from high-level features and is firm against difficult lighting, blurredness, and different camera instincts [14]. The block diagram is shown in Fig. 2.4.1 depicts the underlying architecture of the PoseNet. It subsumes a feature extractor and a regression subnetwork. The feature extractor can be a pretrained DCNN like ResNet34, VGG16, or AlexNet. The regression subnetwork consists of an average pooling, dropout, a fully connected layer networked sequentially. It receives the high dimensional features from the feature extractor. Through the average pooling and dropout layers, it is then reduced to a lower dimension for generalization and faster computation in the PoseNet model [19].

These poses are in 6-Degree of Freedom (DoF), which define the six parameters in translation and rotation [14]. The translation parameters include forward/backward ($x-axis$), left/right ($y-axis$), and up-down ($z-axis$). The rotation parameters include yaw ($normal-axis$), pitch ($transverse-axis$), and roll ($longitudinal-axis$). These are the core parameters that help in getting the accurate position of the vehicle and then form the trajectory.

But these six parameters are converted to 7 coordinates as seen in the Fig. 2.4.1 where x_1, x_2, and x_3 are translation coordinates, whereas y_1, y_2, y_3, y_4 are the rotation

coordinates. This anomaly happens when reading the input rotation poses from the file as euler angles, but during pre-processing stages, euler angles are converted to quaternions. Quaternions are the set of four values (x_o, y_1, y_2 and y_3) where x_o represents a scalar rotation around vector - y_1, y_2 and y_3. Function mat2quat from transforms3d.quaternions package performs the quaternion conversion, which takes parameters and returns the output as following:
1. Parameters: array-like 3×3 rotation matrix.
2. Returns: (4,) array closest quaternion to input matrix.

Function mat2quat performs the conversion of euler angles to quaternions as given in the equations below:

$$x_0 = (\sqrt{1 + c_1c_2 + c_1c_3 - s_1s_2s_3 + c_2c_3})/2, \tag{2.2}$$

$$y_1 = (c_2s_3 + c_1s_3 + s_1s_2c_3)/4x_0, \tag{2.3}$$

$$y_2 = (s_1c_2 + s_1c_3 + c_1s_2s_3)/4x_0, \tag{2.4}$$

$$y_3 = (-s_1s_3 + c_1s_2c_3 + s_2)/4x_0, \tag{2.5}$$

where $c_1 = \cos(roll/2)$, $c_2 = \cos(yaw/2)$, $c_3 = \cos(pitch/2)$, $s_1 = \sin(roll/2)$, $s_2 = \sin(yaw/2)$, and $s_3 = \sin(pitch/2)$.

2.5 Unimodalilty-based Learning

ResNet (18, 34, 50, 101): ResNet18 was introduced to compete in ILSVRC (2015), where it outperformed other feature extractor models like VGG, GoogLeNet and Inception. All the ResNet models are trained on the ImageNet database, which consists of the images defined by ImageNet Large Scale Visual Recognition Competition (ILSVRC). Experiments have depicted that even though ResNet18 is a subspace in ResNet34, yet, its performance is more or less equivalent to ResNet34. ResNet18, 34, 50, 101 consist of 18, 34, 50 and 101 layers respectively. This paper evaluates the performance of the ResNet as mentioned above models besides other feature extractors and consequently chooses the best models for further fusions and hybrid fusion, thereby, establishing a good trade-off between depth and performance. ResNet models constitute of residual blocks where, ResNet18 and 34 have two-layer deep residual blocks, whereas, ResNet50 and 101 have three-layer deep residual blocks. It consists of five convolutional stages, which is followed by an average pooling layer. There is one fully connected layer and, finally, a 1000-way softmax layer to generate the 1000 class labels [20].

VGG (16, 19): Simonayan and Zisserman developed this for the ILSVRC challenge (2014). It stood 2nd in the classification challenge in ILSVRC with the error in top 5 as 7.32%. VGG16 and 19 consist of 16 and 19 convolution layers, respectively, with max-pooling layer after a set of two or three convolution layers. It comprises two fully connected layers and a 1000-way softmax layer. The main drawbacks of VGG models are high training time and high network weights [21].

AlexNet: It was a winner in 2012 ILSVRC with a breakthrough performance. It consists of 60 million parameters and 650, 000 neurons, making it one of the enormous models back in 2012. It consists of five convolution layers where the max-pooling layer follows the first and second layers. But, third, fourth, and fifth convolution layers are connected directly, which is followed by one fully connected layer and 1000-way softmax layer. It was the first model in 2012 to adopt rectified linear units instead of tanh and usage of the dropout layer to eradicate overfitting [22].

2.6 Multimodality-based Learning

Multimodality fusion is one of the effective and widely adopted techniques used across various learning systems to improve the performance [23–25]. Whereby, different models or the complementary cues from the models are combined using strategies, such as mathematical modeling and subspace transformation [26].

Many existing researches have taken advantage of various fusion strategies. For an instance, Jinwei*et al.* [26] uses Inception-ResNet-v1 model to extract the features. But since the dataset used for the work was too small and just comprised of 15 classes, therefore, Inception-ResNet-v1 was modified to alleviate the overfitting. Six convolutional layers in the model are reduced down to four layers, followed by the addition of the fully connected layer in order to reduce the output to 64. The last layer, i.e., the softmax layer generates the class labels only for the 15 classes.

Akilan *et al.* [25] employs the transfer learning technique in the feature fusion study as the weights from the pre-trained models are copied and used in the first n layers of the target network. The feature extractors used for the work are AlexNet, VGG16, and Inception-v3. As these extractors will result in the feature space of varied dimensions and space, feature space transformation and energy-level normalization were performed to embed the features into a common sub-space using dimensionality reduction techniques like PCA. Finally, the features are fused using fusion rules like concatenation, feature product, summation, mean value pooling, and maximum value pooling.

Research work by Yun Fu *et al.* [27] also uses dimension normalization techniques to produce the consistently uniform dimensional feature space.It presents a sub-space learning method for dimension reduction and multiple feature fusion that is used through supervised and unsupervised learning. Besides normalization strategies, this paper also introduced a new technique called Tensor-Based discriminative sub-space learning. This technique gives better results as it produces the final fused feature vector of adequate length, i.e., the long vector if the number of features is too large and the shorter vector if a number of features are small. Soheil *et al.* [28] introduces a multimodal task-driven dictionary learning algorithm for information that is obtained either homogeneously or heterogeneously. These multimodal task-driven dictionaries produce the features from the input data for classification problems.

Referring to the researches mentioned above, two types of multimodality fusion are performed in this project work, which is as –

1. Early fusion via layer weight enhancement
2. Late fusion via score refinement

Early fusion involves fusing Fully Connected (FC) layer weights of ResNet101 and VGG19 unimodals using addition and multiplication mathematical operation followed by updating the dense layer in either of the unimodal. Whereas, late fusion focuses on fusing the predicted scores after testing the five unimodals on the validation test dataset using the average mathematical operation.

2.7 Hybrid Learner

The hybrid fusion performed in this research work makes it more unique and insightful than the existing hybrid fusion works. It focuses on enhancing and updating the weights of the pretrained unimodals (ResNet101 and VGG19) before training the model, which is followed by fusing the predicted scores obtained in the testing phase. Thus, hybrid fusion is the summation of early and late fusion under a single network architecture.

Sun *et al.* [29] proposes a hybrid convolutional neural network for face verification in wild conditions. In this paper, a new approach is proposed wherein, feature extraction and recognition are combined in one architecture. Instead of extracting the features separately from the images, the features from the two images are jointly extracted by filter pairs. The extracted features are then processed through multiple layers of the deep convolutional neural network to extract high-level and global features. The higher layers in the deep convolutional neural network discussed in the above research paper locally share the weights, which is quite contrary to conventional CNNs. In this way, feature extraction and recognition are combined under the hybrid model.

Similarly, Pawar *et al.* in [30], developed an efficient hybrid approach involving invariant scale features and machine learning techniques used for object recognition. In the feature extraction phase, the invariant features like color, shape, texture, etc., are extracted and subsequently fused together to improve the recognition performance. The fused feature set is then fed to the pattern recognition algorithms, such as, Support Vector Machines (SVMs), Discriminant Canonical Correlation, and Locality Preserving Projections, which likely produces either three distinct or identical numbered false positives. To hybridize the process entirely, a decision module is developed using Neural Networks (NNs) that takes in the match values from the chosen pattern recognition algorithm as input and then returns the result based on those match values. As seen from the related works mentioned above, our approach is quite different from the existing methods as it does not only fuse the weights in the dense layer but also fuses the predicted scores in the testing phase.

2.8 Loss Function

The regressor subnetwork is trained to minimize the translation and rotation errors. The objective of this training is governed by the single loss function (L_β) as given in (2.6):

$$L_\beta(I) = L_x(I) + \beta L_q(I), \tag{2.6}$$

where L_x, L_q are the losses of translation and rotation respectively, and I is the input visual. β is a scaling factor that is used to balance both the losses [31]. β is calculated using homoscedastic uncertainty that combines the losses as defined in (2.7):

$$L_\sigma(I) = \frac{L_x(I)}{\hat{\sigma}_x^2} + \log \hat{\sigma}_x^2 + \frac{L_q(I)}{\hat{\sigma}_q^2} + \log \hat{\sigma}_q^2, \tag{2.7}$$

where $\hat{\sigma}_x$ and $\hat{\sigma}_q$ are the uncertainties for translation and rotation respectively. Here, the regularizers $\log \hat{\sigma}_x^2$ and $\log \hat{\sigma}_q^2$ prevent the values from becoming too big [31]. The loss while training the model can be calculated using a stable derivation as shown in (2.8):

$$L_\sigma(I) = L_x(I)^{-\hat{s}_x} + \hat{s}_x + L_q(I)^{-\hat{s}_q} + \hat{s}_q, \tag{2.8}$$

where the learning parameter $s = \log \hat{\sigma}^2$. The values $\hat{s_x}$ and $\hat{s_q}$ are set to $\hat{s_x} = 0$ and $\hat{s_q} = -3.0$ to get the best results according to [31].

2.9 Evaluation Metric

Measuring the performance of the machine learning model is pivotal to comparing the various CNN models. Since every CNN model is trained and tested on different datasets with varied hyperparameters, it is necessary to choose the right evaluation metric. Choosing the right evaluation metrics depends on the type of model as well, i.e., if it is a regression model or a classification model. Alexei Botchkarev in [32] provides an overview of various performance metrics and their contribution to choosing the right CNN model. It divides metrics under primary metrics, extended metrics, composite metrics, and hybrid metrics. Primary metrics comprise of Mean Absolute Error (MAE), Mean Squared Error (MSE), etc. Whereas extended metrics consist of Normalized Root Mean Squared Error (NRMSE), and composite metrics consist of Mean Absolute Scaled Error (MASE), Relative Mean Absolute Scaled Error (RMAE). Hybrid metrics means when two or more performance metrics are used simultaneously in the same experiment.

In our project work, Mean Absolute Error (MAE) has been used to measure the performance of the set of models ranging from unimodals to hybrid fusion models. MAE is a linear score which is calculated as an average of the absolute difference between the target variables and the predicted variables. It is calculated using the formula given in the equation given below:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} \mid x_i - x \mid \tag{2.9}$$

Since, it is an average, therefore, it implies that all the distinct values are weighted equally in the average.

Chapter 3

Unimodality-based Learning

3.1 Overview

Unimodality-based learning is one of the most common and accepted techniques for machine learning implementations. It acts as the building block of the multimodality and hybrid learning as each unimodal combines to form a multimodal. The unimodal-based learning approach leads to an independent learning strategy wherein all the models are independent of each other and are solely responsible for the performance obtained. The unimodal approach in our project uses seven different types of DCNNs, and these DCNNs act as a distinct feature extractor thereby, giving different results when passed on the same vision model, i.e., PoseNet.

3.2 Unimodals

3.2.1 ResNet

ResNet is the most potent DCNN developed in Microsoft research to ease network training, which is significantly deeper than formerly used networks in 2015 [33]. It has won ILSVRC 2015 challenge by achieving fabulous performance results on ImageNet 1000-way classification dataset with a depth of up to 152 layers through learning the residual representation functions. It has also achieved excellent results in the generalization of other identification tasks and won first place on the COCO dataset in COCO 2015 competitions. ResNet solves the problem of exploding or vanishing gradient by adding the input to the output after a few weight layers in network architecture. If there is a vanishing gradient in the output of weight layers, the network always has the identity input to backpropagate to earlier layers.

ResNet architecture has several variants that have the same definition but with a different number of layers. The term ResNet accompanied by two or more digits clearly shows the architecture of ResNet with a certain number of neural network layers. In our project we have used ResNet18, ResNet34, ResNet50, and ResNet101.

1. Architecture of ResNet

A preprocessed 224×224 sized image given as input for example, to the ResNet model consists of one single stage of convolution and pooling accompanied by four layers of identical behavior. This is followed by a 3×3 convolution with a fixed dimensional feature map (F) [64, 128, 256, 512], respectively, followed by batch normalization and activation function after each convolution [33]. After every two convolutions, a shortcut connection is inserted, forming a 2 layer bottleneck design to residualized the network (For ResNet18 and ResNet34). The identity shortcuts can be explicitly used when input and output are of the same dimension, but when the dimension increases, network has two options for consideration:

1. The shortcut still performs identity mapping, with padded additional zero entries for increasing dimensions.
2. A projection shortcut is used to match dimensions. Average pooling is performed to the output of convolutions, which is followed by a 1000-way softmax layer.

In the architecture of ResNet50, a shortcut connection is inserted after every three convolution layers forming a 3-layer bottleneck design. For the construction of ResNet101 architecture, another 3-layer bottleneck design is being used.

3.2.2 VGG

K.Simonyan and A.Zisserman proposed VGG ConvNet at the Visual Geometry Group lab of Oxford Robotics Institute in 2014. VGG had achieved 92.7% top-5 accuracy on the ImageNet dataset, which contains 14 million images belonging to 1000 classes [21]. A variety of structural variants of VGG have been developed and tested, but VGG16 and VGG19 are most frequently referred because of their accuracy and depth. It has 138 million parameters. On behalf of the number of layers and architecture structure, a variety of VGG variants are named. The most commonly used VGG models are VGG16 and VGG19 for 16 and 19 learned layers, respectively.

1. VGG16

VGG16 is the variant of the VGG convolutional neural network, which comprises 16 layers. The architecture of VGG16 consists of 13 convolutional layers, two fully-connected layers, and one soft-max classifier layer [21]. Convolutional layers are divided into five sets, followed by a max-pooling layer. Each set of convolutional layers has a different number of feature kernels of size 3×3. RGB image of fixed size 224×224 is passed as input to the first set of convolutional layers. The first set comprises first and second layers with 64 feature kernels. The resulting output from the first set of convolutional layers is passed to a max-pooling layer with a stride rate of 2. The second set contains the third and the fourth convolutional layer with 128 kernels and a max-pooling layer with a stride rate of 2.

The output from the max-pooling layer reduced to $56 \times 56 \times 128$. Fifth, sixth, and seventh convolutional layers have 256 feature kernels filters in each layer. The output from the seventh convolutional layer is fed into the max-pooling layer with a stride rate of 2. The fourth and the fifth set of convolutional layers contains three convolutional layers in each with 512 kernel feature maps, followed by a max-pooling layer with a stride rate of 1 after tenth and thirteenth convolutional layer respectively. Fourteenth and fifteenth layers are the two fully-connected layers which have 4096 channels in each layer. The final layer is the 1000-way soft-max layer for the classification of 1000 different classes. All the hidden layers equip rectified Linear Units (ReLU) non-linearity.

2. VGG19

VGG19 is a variant of VGG ConvNet that consists of 19 layers, i.e., 16 convolutional layers, two fully-connected layers, 5 MaxPool layers, and 1 SoftMax layer. RGB image of fixed size is given as input to the network of shape $224 \times 224 \times 3$. Each convolutional layer has a certain number of kernels sized 3×3 with a stride size of 1 pixel. VGG19 has the same layer structure as VGG16, except fouth and fifth convolutional layer set consists of 4 convolutional layers with 512 kernels each layer.

3.2.3 AlexNet

Inspired by the Y. LeCun developed neural network LeNet, AlexNet is designed and developed by [22]. AlexNet was the largest neural network which was designed to address ILSVRC-2010 competition for the classification of object images into one of $1,000$ different categories. It contains 60 million parameters and 650K neurons. AlexNet model was firstly trained with data augmentation, by artificially increasing the size of the training dataset ImageNet.

The architecture of AlexNet is divided into two different parts:

1. Feature extraction – It contains five convolutional layers.
2. Classifier – It includes two fully connected layers and one soft-max layer.

The first and second convolutional layer performs convolutional tasks and max-pooling with Local Response Normalization, with the kernel size of 96 and 256 respectively. Pooled and normalized output of the second convolutional is fed into a third convolutional layer, which is connected to the fourth and fifth convolutional layer without any pooling and normalization in between the layers with the kernel size of 384, 384, and 296 respectively. The following convolutional layers are followed by two fully connected layers with a dropout layer to improve generalization error. Each fully-connected layer has 4096 neurons. Rectified Linear Units are applied to the output of each convolutional and fully-connected layer. The output of a fully-connected layer is passed to a 1000-way soft-max layer to produce a distribution over the 1000 class labels [22].

3.3 Conclusion

The unimodal-based learning approach addresses the experimental study of seven pretrained feature frameworks which are trained and tested on the dataset by passing through PoseNet to get translation and rotational errors.

Chapter 4

Multimodality-based Learning

4.1 Overview

Mutimodality fusion is one of the effective and widely adopted techniques used across various learning systems to improve the performance [25, 34, 35]. Whereby, different models or the complementary cues from the models are combined using strategies, such as mathematical modeling and subspace transformation [26]. Multimodality fusion has been performed extensively in the field of medicine and in applications varying from satellite communication to autonomous driving. Multimodality fusion guaranties to stabilize autonomous driving vehicles [36].

Multimodality fusion can be performed in three ways depending on the input. It can be either decision-level, data-level or feature-level. Decision-level performs fusion based on the output of the results obtained. This can also be considered to be a late fusion approach as the fusion happens after the results of the trained model are attained. Decision-level fusion can be performed using the predicted scores after testing. This method is called score fusion or score refinement. However, the data-level and feature-level fusion make use of the features obtained and the source of data to get best results by complex fusion of the different layers of multimodals. This approach can also be referred to as early fusion as this type of fusion happens before training the model.

This work focuses on the performance improvements of a SLAM model using multimodal learning. The first approach being early fusion via layer weight enhancement of feature extractors, and the second is the late fusion via score refinement of the pose regressor.

4.2 Proposed Method

4.2.1 Preprocessing

Before passing the images and poses to the PoseNet model, it is required to preprocess the data adequately. The preprocessing involves checking the consistency of the images, resizing and center cropping of the images, extraction of mean and standard deviation, and normalization of the poses. Checking the consistency of the images involves the checking of the timestamp of images extracted from the two cameras. Consistent images will have the same timestamp in both the cameras for when they were clicked. Inconsistent images are discarded and not considered for further processing. The images are resized to 260 × 260 and center cropped to 250 × 250. This ratio has been used for both the train and the test samples to maintain a consistency for the images used. From the ground truth of translation values, the minimum, maximum, mean, and standard deviation are computed. The rotation values are read as Euler angles which suffer from wrap around infinities, gimbal lock and interpolation problems. To overcome these challenges, Euler rotations are converted to seven quaternions [37].

4.2.2 Multimodality-based Learner Models

The proposed solution introduces two strategies to enhance the performance:

1. Early fusion via layer weight enhancement of feature extractor
2. Late fusion via score refinement of the trajectory (pose) regressor

The former fuses the weights of the features extracted; the latter fuses the modalities at the score level.

1. Early fusion via layer weight enhancement (E_{fusion})

The operational flow of the early fusion via layer weight enhancement during training is shown in Fig. 4.2.1. ResNet101 and VGG19 are selected as the feature extractors. These two models are selected based on their individual performances on the Apolloscape test dataset.

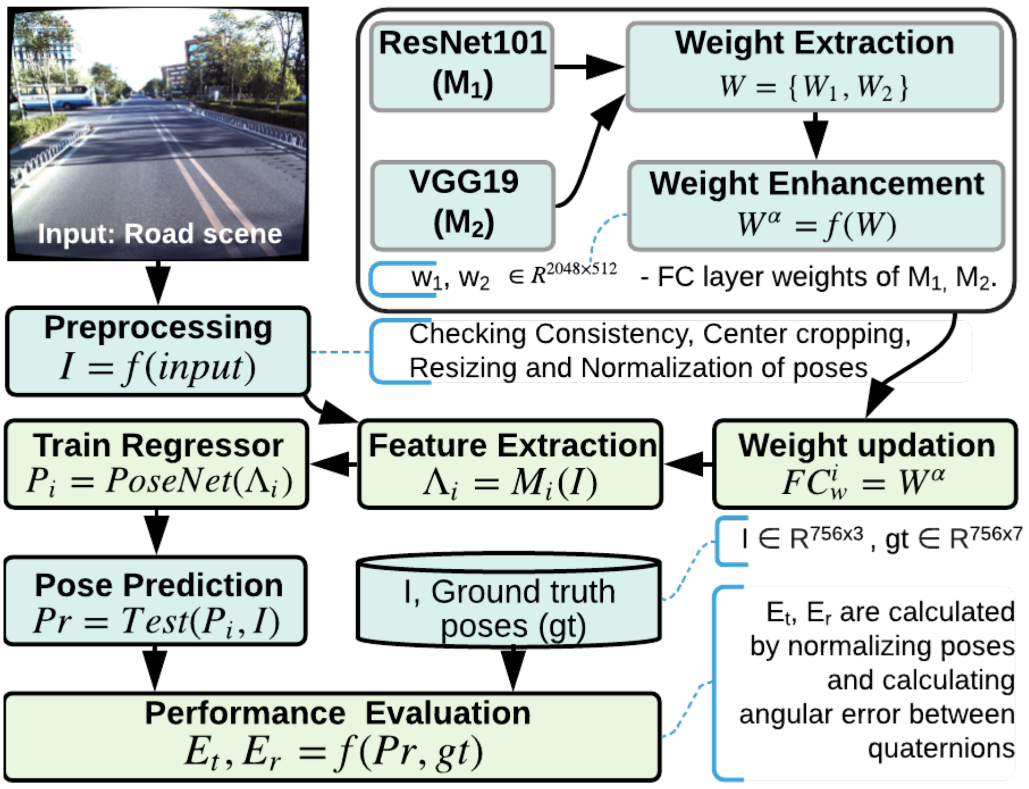
Input: Road scene
ResNet101 (M1)
VGG19 (M2)
Weight Extraction $W = \{W_1, W_2\}$
Weight Enhancement $W^\alpha = f(W)$
$W_1, W_2 \in R^{2048\times512}$ - FC layer weights of M1, M2.
Preprocessing $I = f(input)$
Checking Consistency, Center cropping, Resizing and Normalization of poses
Train Regressor $P_i = PoseNet(\Lambda_i)$
Feature Extraction $\Lambda_i = M_i(I)$
Weight updation $FC^i_w = W^\alpha$
$I \in R^{756x3}$, $gt \in R^{756x7}$
Pose Prediction $Pr = Test(P_i, I)$
I, Ground truth poses (gt)
Et, Er are calculated by normalizing poses and calculating angular error between quaternions
Performance Evaluation $E_t, E_r = f(Pr, gt)$

Figure 4.2.1: Early fusion via feature extractor weight enhancement.

The weights from the Fully Connected (FC) layer are extracted from each model, viz., VGG19 and ResNet101 as shown in the equations (4.1) and (4.2) respectively.

$$FC_{V\rightarrow VGG19} = f\left(\sum_{i=1}^{N} w_i^V x_i + b^V\right), \tag{4.1}$$

where $f(\cdot)$, N, i, w_i^V, x_i and b^V are activation function, total number of neurons in the FC_V layer, neuron index, weight of i^{th} neuron of FC_V layer, input to the FC_V Layer, and bias of VGG19's FC layer respectively.

$$FC_{R\rightarrow ResNet101} = f\left(\sum_{i=1}^{N} w_i^R x_i + b^R\right), \tag{4.2}$$

where $f(\cdot)$, N, i, w_i^R, x_i, and b^R are activation function, total number of neurons in the FC_R layer, neuron index, weight of i^{th} neuron of FC_R layer, input to the FC_R layer, and bias of ResNet101's FC layer respectively. The FC layer weights of the two feature extractors are fused by using arithmetic operations as defined in the equation (4.3).

$$\hat{W} = f(W^V, W^R), \tag{4.3}$$

where $f(.)$, W^V, W^R are operations like arithmetic addition or multiplication, weight of the FC layer of VGG19 and weight of the FC layer of ResNet101 respectively. The fused values are used to update the weights of FC layer of the ResNet101 feature extractor as given in equation (4.4).

$$FC_m = f\left(\sum_{i=1}^{N} \hat{W}_i x_i + b^m\right), \tag{4.4}$$

where FC_m, N, $\hat{W}_i$, x_i, and b^m denote the updated dense layer, total number of neurons in the updated dense layer, weight of i^{th} neuron of updated model, input of dense layer and bias for the updated model respectively. The visual representation of the extracted and enhanced weights of the two models are shown in figures 4.2.2 and 4.2.3.

The trained model is further tested on the test dataset to retrieve the predicted poses. These predicted poses are compared with the ground truth for performance evaluation. The rotation loss is calculated by the quarternion angular error of the quaternion angles in the predicted poses. The translation loss is calculated by the normalization of poses in the x, y and z coordinates. The mean and the median of the translation and rotation errors are computed for comparison. However, the inference time taken during early fusion is more than that of unimodals run on the same architecture.

A. Training steps for early fusion

Step 1: The weights from the fully connected (FC) layers of the pretrained DCNNs (ResNet101 and VGG19) used, are extracted as given by equations (4.1) and (4.2).

Step 2: The weights extracted in Step 3 are enhanced by mathematical operations like addition or multiplication to get the final updated weight by equation (4.3).

Step 3: The final weight is updated to the FC layer of either ResNet101 or VGG19 as shown in equation (4.4).

Step 4: The pretrained model with updated weights is trained on the training dataset using the designed DCNN PoseNet architecture.

Step 5: Using batch size 34, learning rate 0.01, drop out rate 0.5, and Adam is used as the optimizer, the PoseNet architecture is trained for 1 epoch.

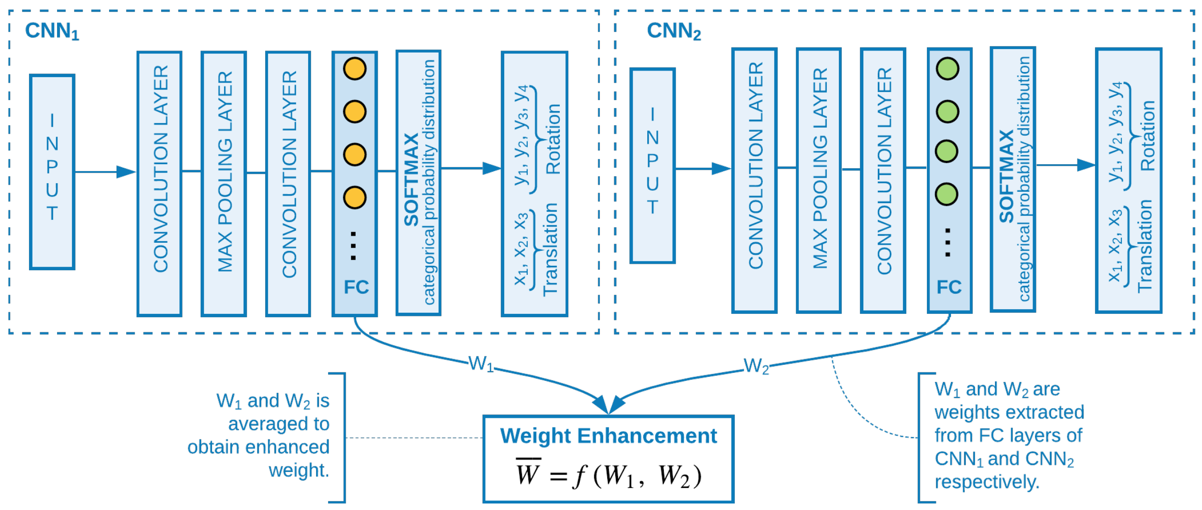
CNN1
INPUT
CONVOLUTION LAYER
MAX POOLING LAYER
CONVOLUTION LAYER
FC
SOFTMAX categorical probability distribution
x1, x2, x3 Translation
y1, y2, y3, y4 Rotation
CNN2
W1
W2
W1 and W2 is averaged to obtain enhanced weight.
Weight Enhancement
$\overline{W} = f(W_1, W_2)$
W1 and W2 are weights extracted from FC layers of CNN1 and CNN2 respectively.

Figure 4.2.2: Weight enhancement of two CNN models in early fusion.

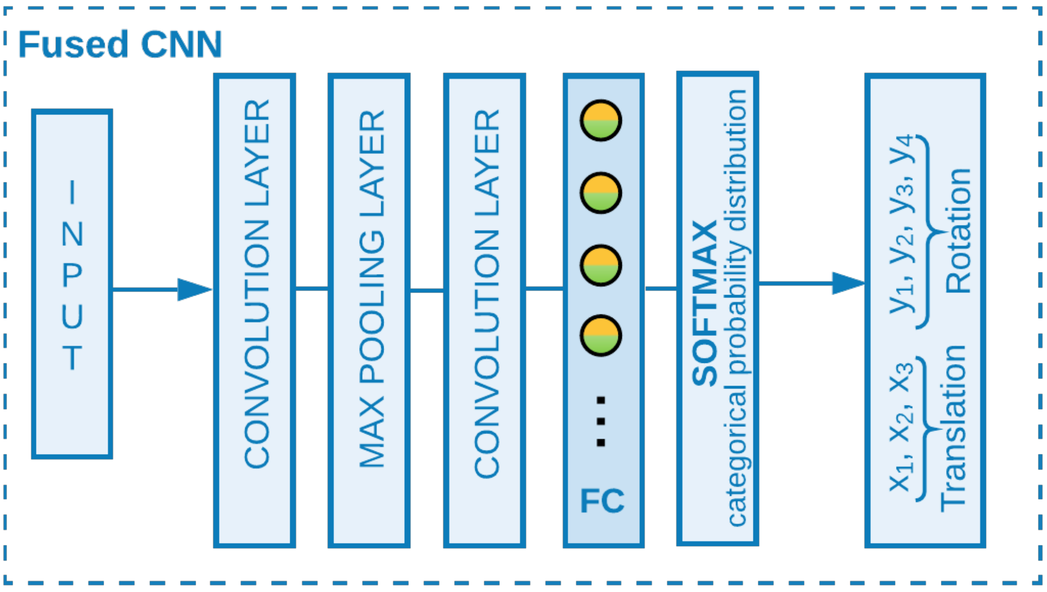

Figure 4.2.3: Updated model with enhanced weights.

Step 6: Use step 5, 999 times to obtain the final trained DCNN model.

B. Inference steps for early fusion

Step 1: Use the trained DCNN model to test on the validation dataset.

Step 2: The images are regressed by the trained DCNN model to obtain the predicted poses.

Step 3: Comparison of the predicted poses with the ground truth poses to calculate the translation and rotation losses as shown in equations (2.6), (2.7), and (2.8).

Step 4: The mean and the median of the translation and rotation errors are computed for comparison.

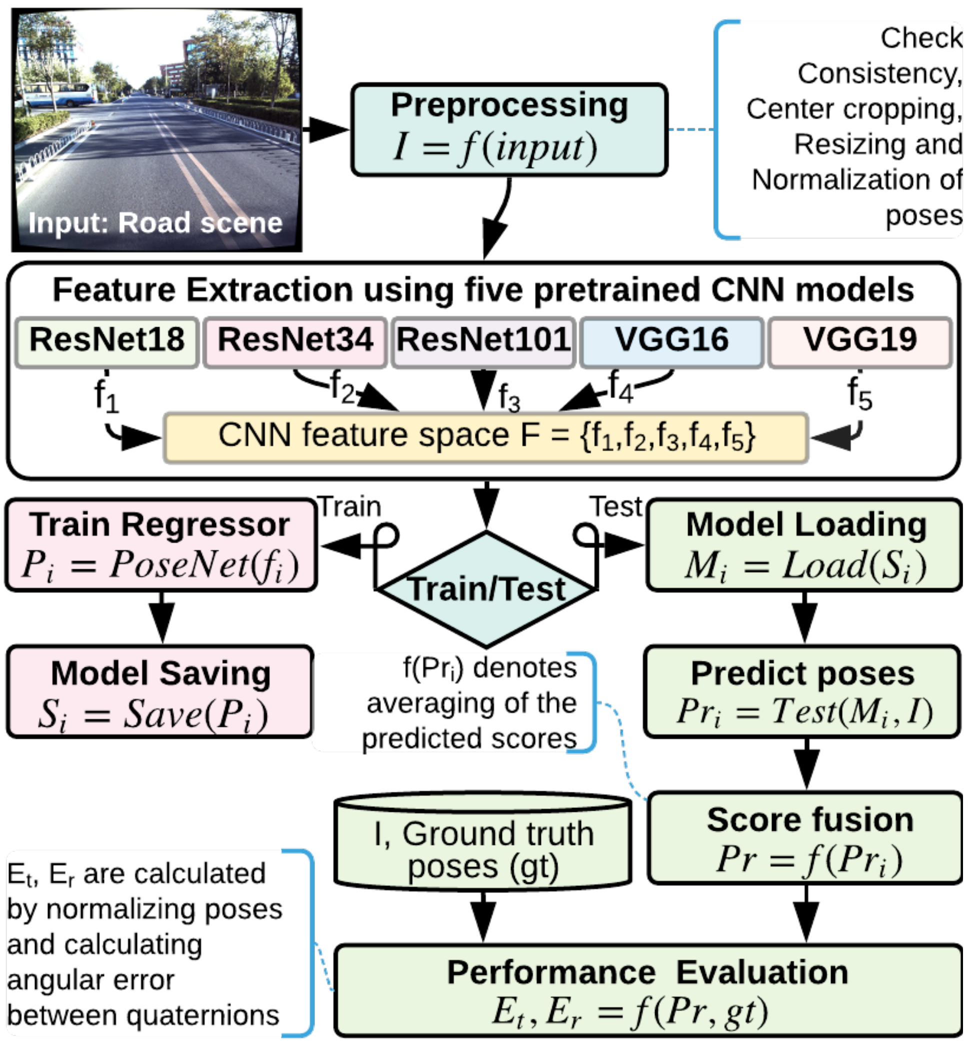
Input: Road scene
Preprocessing
$I = f(input)$
Check Consistency, Center cropping, Resizing and Normalization of poses
Feature Extraction using five pretrained CNN models
ResNet18
ResNet34
ResNet101
VGG16
VGG19
f_1
f_2
f_3
f_4
f_5
CNN feature space F = {f_1,f_2,f_3,f_4,f_5}
Train
Test
Train Regressor
$P_i = PoseNet(f_i)$
Train/Test
Model Loading
$M_i = Load(S_i)$
Model Saving
$S_i = Save(P_i)$
f(Pr_i) denotes averaging of the predicted scores
Predict poses
$Pr_i = Test(M_i, I)$
I, Ground truth poses (gt)
Score fusion
$Pr = f(Pr_i)$
E_t, E_r are calculated by normalizing poses and calculating angular error between quaternions
Performance Evaluation
$E_t, E_r = f(Pr, gt)$

Figure 4.2.4: Late fusion model via score refinement.

2. Late fusion model (L_{fusion})

Figure 4.2.4 elaborates the multimodality-based late fusion via score refinement approach. The camera feed and the poses are preprocessed as described in subsection 4.2.1. Then, the pretrained CNN models are used to extract the features from the preprocessed data. In this work, the following five pretrained CNN models ResNet18, ResNet34, ResNet101, VGG16, and VGG19 are used. These five pretrained models are chosen based on their individual performances. Using the individual feature vectors from the aforesaid pretrained CNNs, five different PoseNet models are trained independently.

Then, pose predictions are carried out using trained PoseNet models, $M_{i=\{1,\cdots,5\}}$ simultaneously as shown in (4.5).

$$Pr_i = M_i(I), \tag{4.5}$$

where Pr_i, M_i and I denote the predicted poses, the i^{th} trained PoseNet model, and the test input visual respectively. The predicted values in (4.5) are fused to achieve the refined poses of translation and rotation as

$$Pr = f\left(Pr_i\right), \tag{4.6}$$

where Pr is the distilled poses after late fusion using the average arithmetic operation $f(.)$.

Mathematical operation averaging is performed on all the five predicted poses to get the final predicted score. The final predicted poses are compared with the ground truth poses to get the translation and rotation errors. The mean and the median of the translation and rotation errors are extracted for comparison. However, the average processing time for the score fusion is more in comparison to both the unimodals and the early fusion model.

A. Training steps for late fusion

Step 1: Five best pretrained models – ResNet18, ResNet34, ResNet101, VGG16, and VGG19 are used to extract features on the preprocessed images.

Step 2: The pretrained models are fed to the PoseNet architecture to extract features for regression.

Step 3: Using batch size 34, learning rate 0.01, drop out rate 0.5, and Adam is used as the optimizer for 1 epoch.

Step 4: Use Step 3, 999 times to obtain the 5 trained DCNNs.

Step 5: The five trained DCNN models are saved.

B. Inference steps for late fusion

Step 1: The five saved models are loaded and tested on the validation dataset to obtain the predicted poses as shown in equation (4.5).

Step 2: The five predicted poses are fused using average mathematical operation to get the final predicted scores by equation (4.6).

Step 3: The final predicted score is compared with the ground truth poses to calculate the translation and rotation losses by equations (2.6), (2.7), and (2.8).

Step 4: The mean and the median of the translation and rotation losses are computed for comparison.

4.2.3 Conclusion

This research proposes two multimodal fusion strategies to improve the localization accuracy of a SLAM model which are as follows: early fusion method via layer weight enhancement, and late fusion via score refinement. The aim of introducing two multimodal fusion methods is to improve the performance of the unimodal approaches. The results of the unimodal and the multimodal approaches are evaluated and compared in chapter 6.

Chapter 5

Hybrid Learner

5.1 Overview

Hybrid fusion is an incremental work to improve the accuracy of the Multimodality fusion models. Hybrid fusion has been performed to improve the performance of multimodality-based learners, as discussed in Chapter 4.2.2. It is usually performed on feature-level fusions [38] and score fusions [39]. Hybrid fusion has been used in multiple applications ranging from fingerprint analysis, genetic DNA sequencing, and SLAM.

This work involves a combination of early and late fusion models. The predicted scores of the four early fusion models obtained in multimodality fusion are combined together in hybrid fusion using the average mathematical operation to minimize the translation and rotation losses.

5.2 Hybrid Learner Model

This hybrid learner is the combination of the two methods explained in section 4.2.2. The two combined strategies are (i). early fusion via layer weight enhancement of feature extractor, and (ii). late fusion via score refinement of the trajectory (pose) regressor.

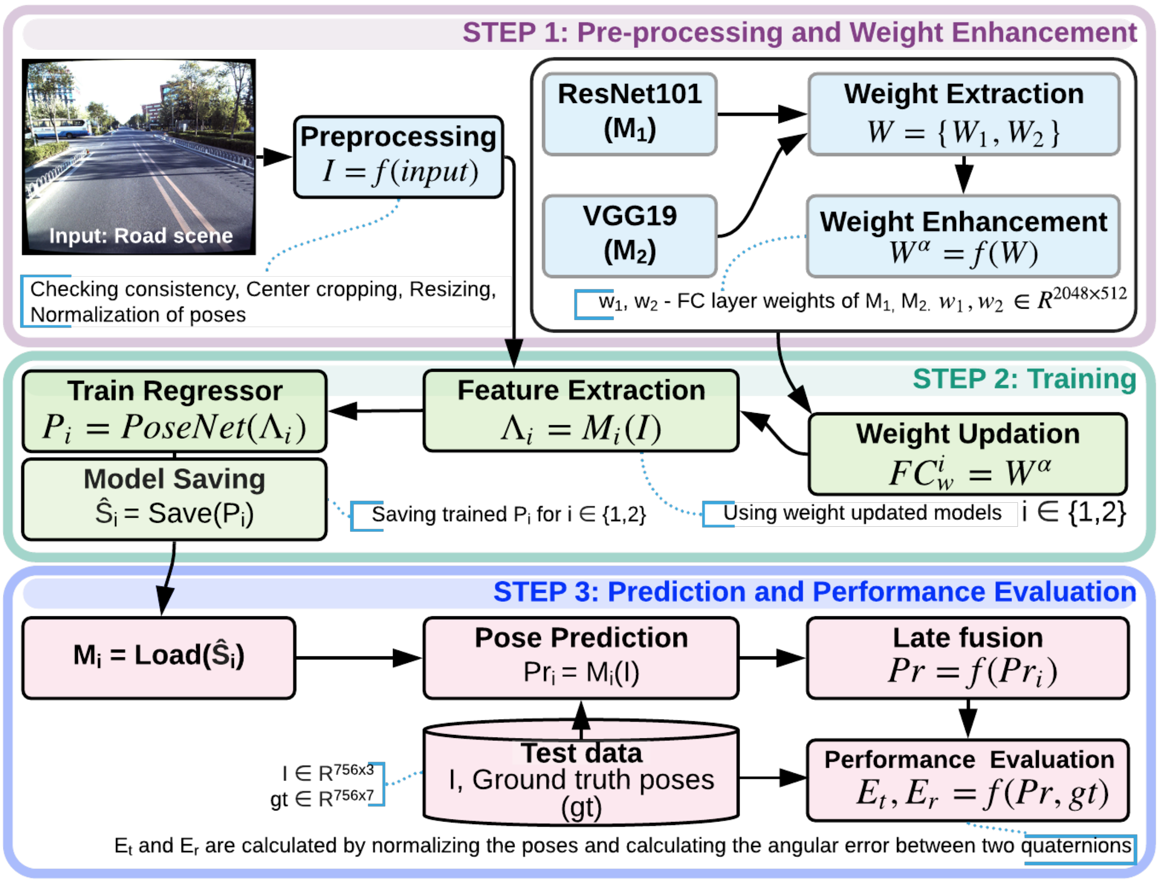
STEP 1: Pre-processing and Weight Enhancement
Input: Road scene
Preprocessing $I = f(input)$
Checking consistency, Center cropping, Resizing, Normalization of poses
ResNet101 (M_1)
VGG19 (M_2)
Weight Extraction $W = \{W_1, W_2\}$
Weight Enhancement $W^\alpha = f(W)$
w_1, w_2 - FC layer weights of M_1, M_2. $w_1, w_2 \in R^{2048\times512}$
STEP 2: Training
Train Regressor $P_i = PoseNet(\Lambda_i)$
Feature Extraction $\Lambda_i = M_i(I)$
Model Saving $\hat{S}_i = Save(P_i)$
Weight Updation $FC_w^i = W^\alpha$
Saving trained P_i for $i \in \{1,2\}$
Using weight updated models $i \in \{1,2\}$
STEP 3: Prediction and Performance Evaluation
$M_i = Load(\hat{S}_i)$
Pose Prediction $Pr_i = M_i(I)$
Late fusion $Pr = f(Pr_i)$
Test data I, Ground truth poses (gt)
$I \in R^{756x3}$
$gt \in R^{756x7}$
Performance Evaluation $E_t, E_r = f(Pr, gt)$
E_t and E_r are calculated by normalizing the poses and calculating the angular error between two quaternions

Figure 5.2.1: The operational flow of the Hybrid Learner.

5.2.1 Hybrid Learner (H_{fusion})

The figure 5.2.1 shows the detailed flow diagram of the hybrid learner. The two best feature extractors chosen for early fusion are used for hybrid fusion. The early fusion models obtained by fusing the Fully Connected (FC) layer weights of ResNet101 and VGG19 by addition or multiplication are used to perform late fusion. In late fusion, the predicted scores achieved from the regression layer of each of the trained early fusion models are fused using averaging to achieve better results.

Hybrid learner's methodology is divided into three subdivisions. These subdivisions are explained in detail below:

1. Preprocessing and Weight Enhancement

The Hybrid learner's preprocessing steps are same as that of the preprocessing steps in multimodality-based learner. The steps are the same as explained in subsection 4.2.1. The preprocessed data is fed to the feature extractors: ResNet101 and VGG19. These two models are selected based on their individual performance on the Apolloscape test dataset. The Fully Connected (FC) layer weights of the two feature extractors are fused by using addition or multiplication. The fused values are used to update the weights of FC layer of the ResNet101 or VGG19 feature extractor. The updated model is used as new feature extractor for the regressor subnetwork. Then, the updated model is trained in the next phase.

2. Training

The regresser in this phase is trained on the training data to obtain the four different models viz., trained model on ResNet101 for addition (M9), trained model on ResNet101 for multiplication (M10) and the trained models on VGG19 for addition and multiplication (M11 and M12) respectively. These models are then saved for the next phase which includes testing and score fusion.

3. Prediction and Performance Evaluation

The early fusion trained models on ResNet101 and VGG19 using multiplication and addition are saved and loaded in the late fusion phase of the operational flow. The loaded early fusion models are tested on the test visuals to calculate the predicted scores. The predicted scores of the early fusion models (ResNet101 and VGG19) using addition are fused together by performing average mathematical operation to obtain the Hybrid Fusion Model (M13). Similarly, the predicted scores of the early fusion models using multiplication are fused (M14). Finally, the predicted scores of the four early fusion models using addition and multiplication are fused together using average mathematical operation to achieve the predicted scores for the third hybrid fusion model (M15). These predicted poses are then compared with the ground truth poses to calculate the mean and median of translation and rotational errors.

A. Training steps for hybrid learner

Step 1: Two best pretrained models – ResNet101 and VGG19 are chosen based on the results of the unimodals.

Step 2: The weights from the fully connected (FC) layers of the pretrained DCNNs (ResNet101 and VGG19) used, are extracted as given by equations (4.1) and (4.2).

Step 3: The weights extracted in Step 3 are enhanced by mathematical operations like addition or multiplication to get the final updated weight by equation (4.3).

Step 4: The final weight is updated to the FC layer of either ResNet101 or VGG19 as shown in equation (4.4).

Step 5: The pretrained model with updated weights is trained on the training dataset using the designed DCNN PoseNet architecture.

Step 6: Using batch size 34, learning rate 0.01, drop out rate 0.5, and Adam is used as the optimizer, the PoseNet architecture is trained for 1 epoch.

Step 7: Use step 6, 999 times to obtain the final trained DCNN model.

Step 8: The four trained models – Early fusion on ResNet101 using addition, early fusion on ResNet101 using multiplication, early fusion on VGG19 using addition, and early fusion on VGG19 using multiplication are saved.

B. Inference steps for hybrid learner

Step 1: The four saved models are tested on the validation dataset to get the predicted scores by equation (4.5).

Step 2: The predicted scores from the four models are fused using average mathematical operation to get the final predicted scores by equation (4.6).

Step 3: The final predicted poses are compared with the ground truth poses to calculate the losses as shown in equations (2.6), (2.7), and (2.8).

Step 4: The mean and median of the translation and rotation losses are computed for comparison of the accuracies of all the models.

5.2.2 Conclusion

The proposed model's first step was conducted on the unimodal analysis which was the fundamental requirement for fusion techniques. It was followed by the multimodality-based learner approach that gave better results in comparison to the unimodals. In order to achieve better accuracies, a hybrid learner was introduced to combine the multimodality-based learner. The hybrid fused models were obtained by averaging the results of the predicted scores and comparing with the ground truth. The results of the hybrid learner are evaluated and compared in Chapter 6.

Chapter 6

Results

The Apolloscape dataset is run on various DCNNs which include ResNet18, ResNet34, ResNet50, ResNet101, VGG16, VGG19 and AlexNet. The results from these models were extracted on the bases of translation and rotation errors, and the second parameter is timing analysis.

6.1 Translation and Rotation Errors

The performance of these models with PoseNet is compared with unimodal, multimodal, and Hybrid learners, as given in Table 6.1.1, which can be described in three subdivisions. The primary section of the table from M1 to M7 is the outcomes acquired from the pretrained unimodality-based learner. The subsequent section extending from M8 to M12 depicts the performances of five multimodality-based learners. M8 represents late fusion (L_f), M9 (E_f^R+) and M10 (E_f^R*) represent early fusion on ResNet101 as feature extractor with addition and multiplication respectively. M11 (E_f^V+) and M12 (E_f^V*) are the results for early fusion on VGG19 with addition and multiplication respectively. The third section consists of hybrid models where M13, M14, and M15 stand for a hybrid model (H_f^+) which combines the early fusion models (M9 and M11), a hybrid model (H_f^*) which combines the early fusion models (M10 and M12) and a hybrid model (H_f) obtained after averaging the predicted scores of the four early fusion models (M9, M10, M11, and M12) respectively. The results are obtained as the mean and median values of translation and rotation errors. The translation errors are measured in terms of meters(m), while the rotation error is measured in degrees ($^\circ$).

6.1.1 Quantitative Analysis

While performing the analysis on the unimodal PoseNet implementation, it is very apparent from the table 6.1.1 that ResNet101 and VGG19 give the best two outcomes among others. While making the comparison we first compare the unimodal with multimodality learning. It is seen from the table, that the late fusion model shows better performance than unimodality-based learners, but not good as compared to the early fusion model. For our comparative study, we choose ResNet101 as our baseline model because it has the best performance outcomes amongst the other unimodels. The late fusion shows a 66% decrease in the translation's median errors and an 84% decrease in the translation's mean errors when compared to ResNet101 (M4). The median of rotation errors in the late fusion model shows a decrease of 2% and the mean of rotation errors shows a 73% increase when compared to our unimodality baseline model (M4).

Model Name	Median e_t (m)	Mean e_t (m)	Median e_r ($^\circ$)	Mean e_r ($^\circ$)	MAPST (s)
M1 - ResNet18	21.194	24.029	0.778	0.900	0.092
M2 - ResNet34	20.990	23.597	0.673	0.824	0.093
M3 - ResNet50	18.583	20.803	0.903	1.434	0.095
M4 - ResNet101	16.227	19.427	0.966	1.230	0.098
M5 - VGG16	17.150	21.571	1.079	1.758	0.103
M6 - VGG19	16.820	19.935	0.899	1.378	0.111
M7 - AlexNet	46.992	53.004	4.282	7.177	0.108
M8 - L_f	9.763	10.561	0.945	4.645	0.146
M9 - E_f^R+	14.870	18.256	0.673	0.784	0.134
M10 - E_f^R*	14.842	18.013	0.779	0.977	0.131
M11 - E_f^V+	11.047	13.840	0.742	1.024	0.137
M12 - E_f^V*	10.730	14.181	0.756	1.141	0.135
M13 - H_f^+	10.400	12.193	0.828	5.155	0.142
M14 - H_f^*	9.307	11.420	1.206	5.455	0.142
M15 - H_f	7.762	8.829	1.008	4.618	0.144

Table 6.1.1: Performance Analysis in terms of error of Translation (e_t) and Rotation (e_r), Mean Average per sample Processing Time (MAPST) calculated in seconds.

On comparison of the baseline model (M4) with the early fusion model using addition having ResNet101 as a feature extractor (M9), it is seen that there is a 9% decrease in translation's median error and a 6% decrease in translation's mean error. While on the other hand, the median of rotation error shows a 43% decrease and mean of rotation error shows a 41% decrease. The comparison with the early fusion model using multiplication (M10) on VGG19 exhibits a 51% decrease in translation's median error and a 37% decrease in translation's mean error. While comparing the rotation values the drop in median value is by 28% and in the mean value by 8%.

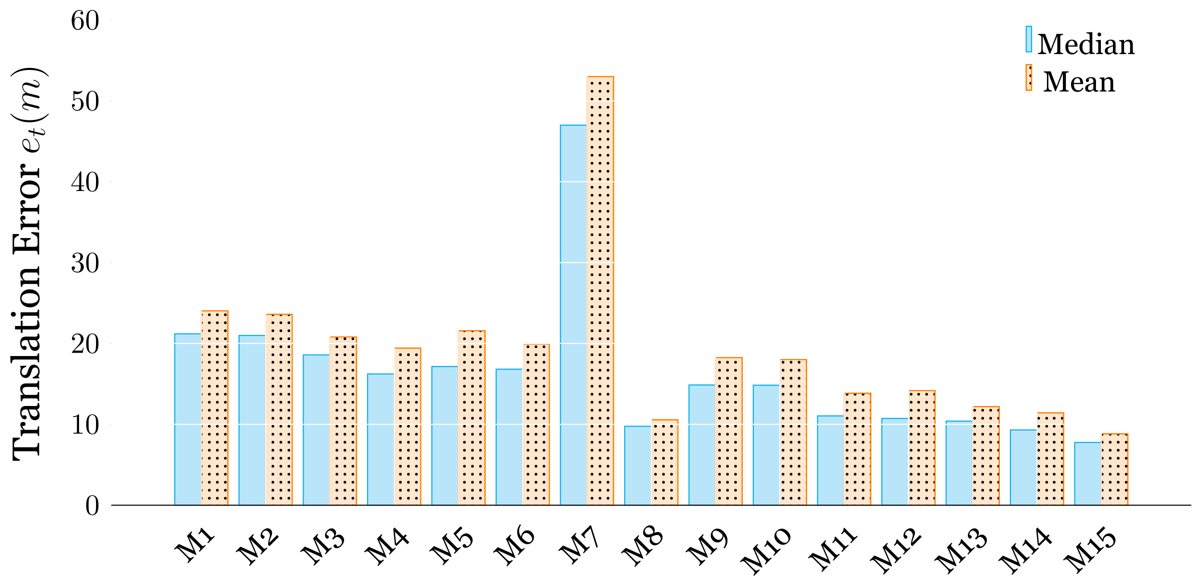

Figure 6.1.1: Error in Terms of Translation.

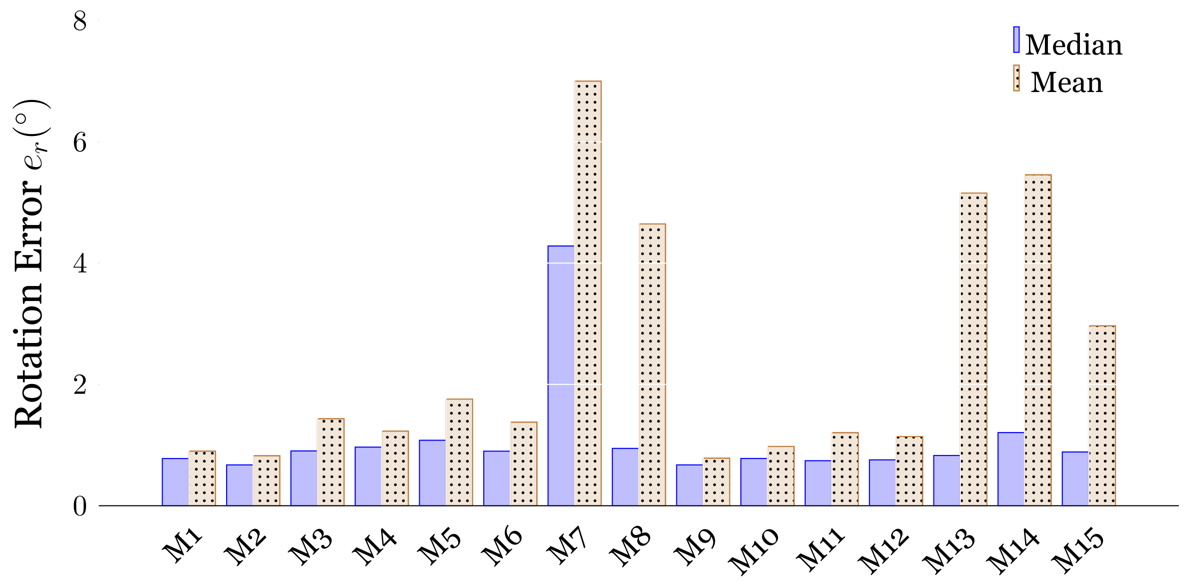

Figure 6.1.2: Error in Terms of Rotation.

It is evident from the table 6.1.1 that hybrid models show much better performance than the unimodal and multimodal ResNet. The hybrid model using averaging shows a 109% decrease in the translation's median errors and a 120% decrease in the translation's mean error when compared to ResNet101. While for rotation errors, there is a 4% increase in the median and a 73% increase in the mean.

6.1.2 Holistic Analysis

On performing the holistic analysis by keeping the RestNet101 unimodal(M4) as the baseline model, it is observed that the late fusion shows an improvement of 37% in translation and 3% in rotation. On considering the early fusion model using ResNet as a feature extractor, the translation shows 6% improvement while rotation shows 31%. The improvement of the early fusion using VGG19 as a feature extractor in terms of translation and rotation is 31% and 24%, respectively. It is quite evident that the hybrid model has bad results for rotation with a decrease of 4% nevertheless, it is the best model considering a huge improvement in translation by 50% among all the experiment results.

Model Name	Improvement in e_t (%)	Improvement in e_r (%)	Timing Overhead (ms)
L_f	37	3	48
E_f (ResNet)	6	31	36
E_f (VGG)	31	24	39
H_f	50	-4	46

Table 6.1.2: Performance Improvement of Proposed Framework when Compared to Baseline Framework ResNet101 with PoseNet.

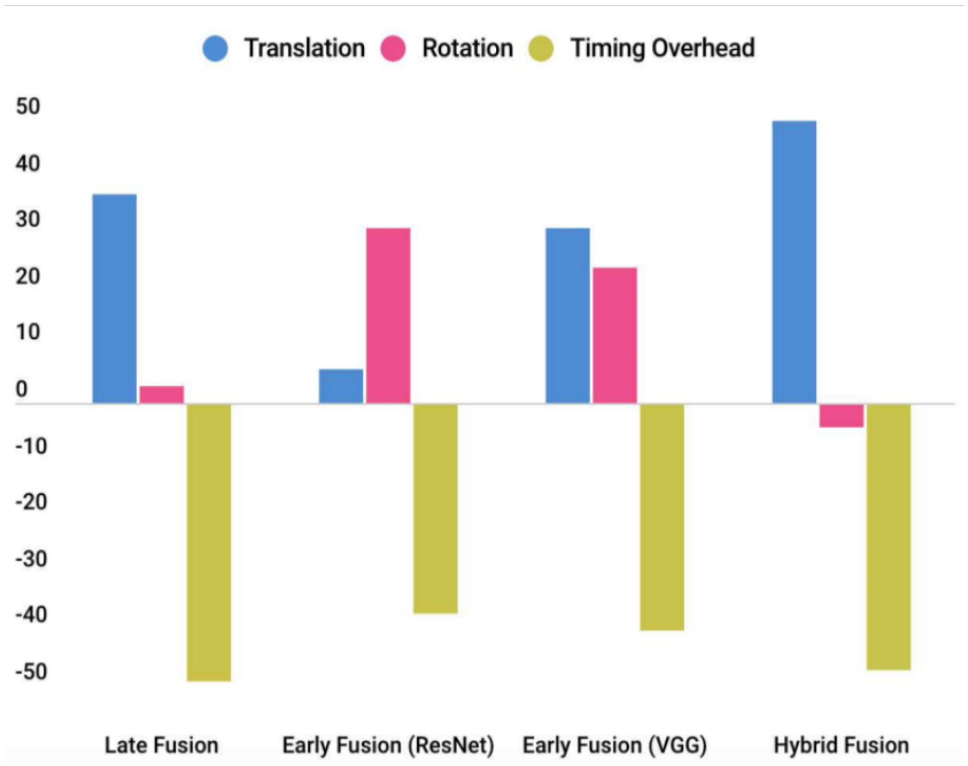

Figure 6.1.3: Holistic Analysis

6.1.3 Qualitative Analysis

The qualitative analysis is estimated by a comparative study of the trajectories obtained. The trajectories imply the visual representation of the ground truth pose values and the predicted pose values, thus helping us to understand the deviation visually. The solid lines in figures denote the ground truth values, and the dots depict the predicted values. We can observe that ResNet101 and VGG19 perform the best in unimodal. In multimodality-based learning, the early fusion model with addition using ResNet101 as feature extractor display better results than others. While considering the hybrid learner, it is quite evident that the hybrid model using averaging gives the best result when compared to the unimodal and multimodal learner, as the deviation is least seen in it.

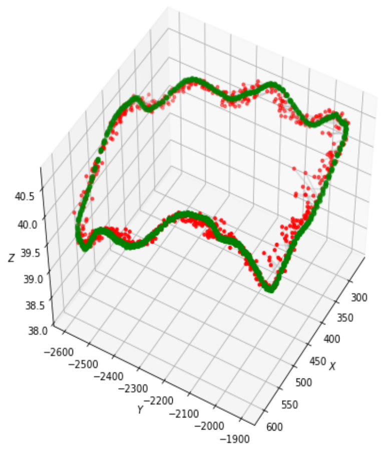

Figure 6.1.4: Trajectory for ResNet34

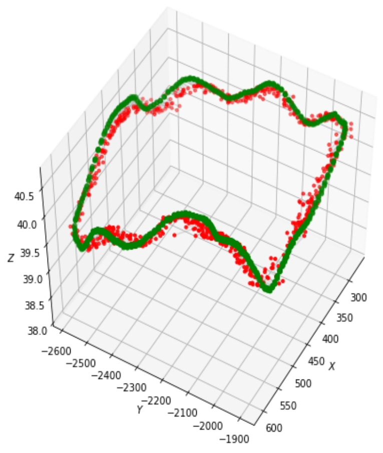

Figure 6.1.5: Trajectory for ResNet34

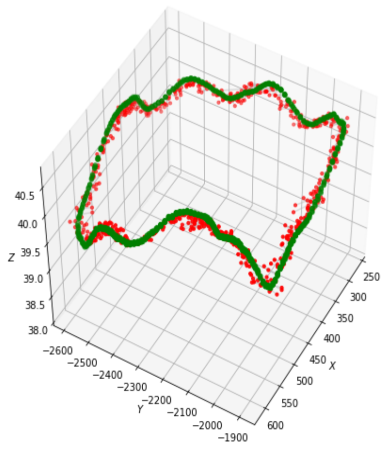

Figure 6.1.6: Trajectory for ResNet101

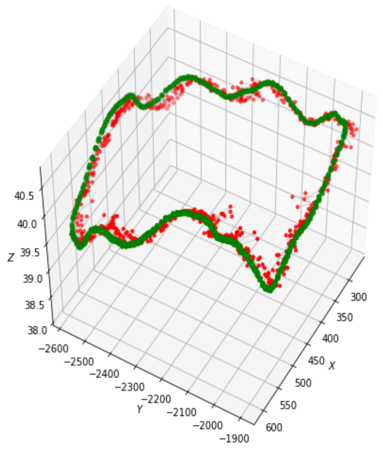

Figure 6.1.7: Trajectory for VGG19

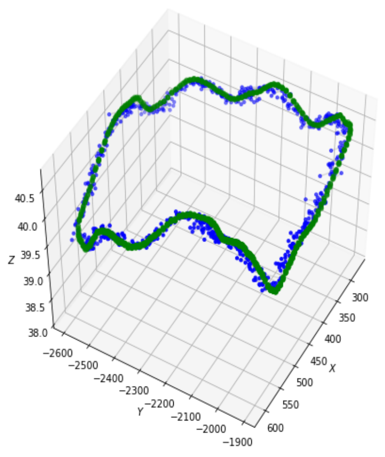

Figure 6.1.8: Trajectory of early fusion using addition with ResNet101 as feature extractor

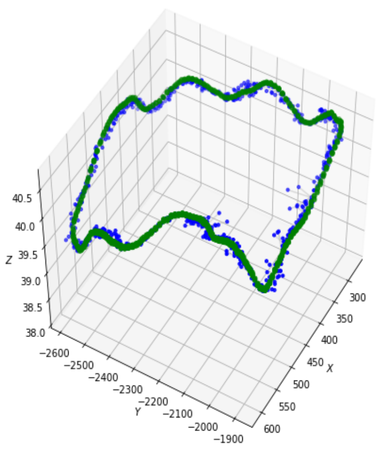

Figure 6.1.9: Trajectory of early fusion using multiplication with VGG19 as feature extractor

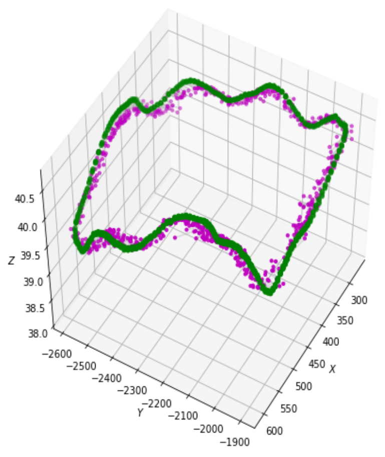

Figure 6.1.10: Trajectory of late fusion

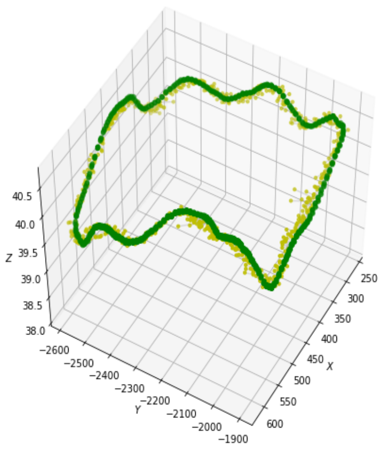

Figure 6.1.11: Trajectory of hybrid fusion with multiplication

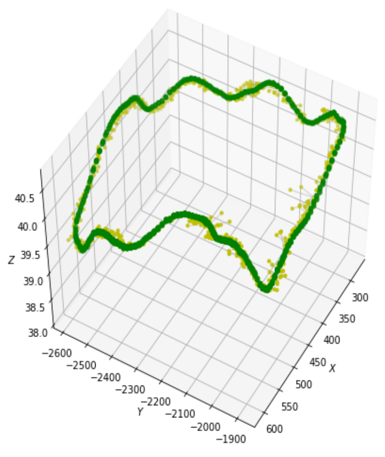

Figure 6.1.12: Trajectory of hybrid fusion with addition

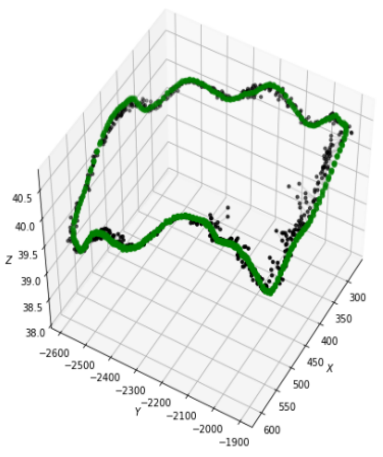

Figure 6.1.13: Trajectory of hybrid fusion with averaging

6.2 Timing analysis

The timing analysis is conducted on a machine that uses an Intel Core i5 processor that uses the Google Colaboratory having a GPU chip Tesla K80 with 2496 CUDA cores, a hard disk space of 319GB and 12.6GB RAM. Table 6.1.1 shows the mean average processing time which is calculated by dividing the inference time with a batch size of 10.

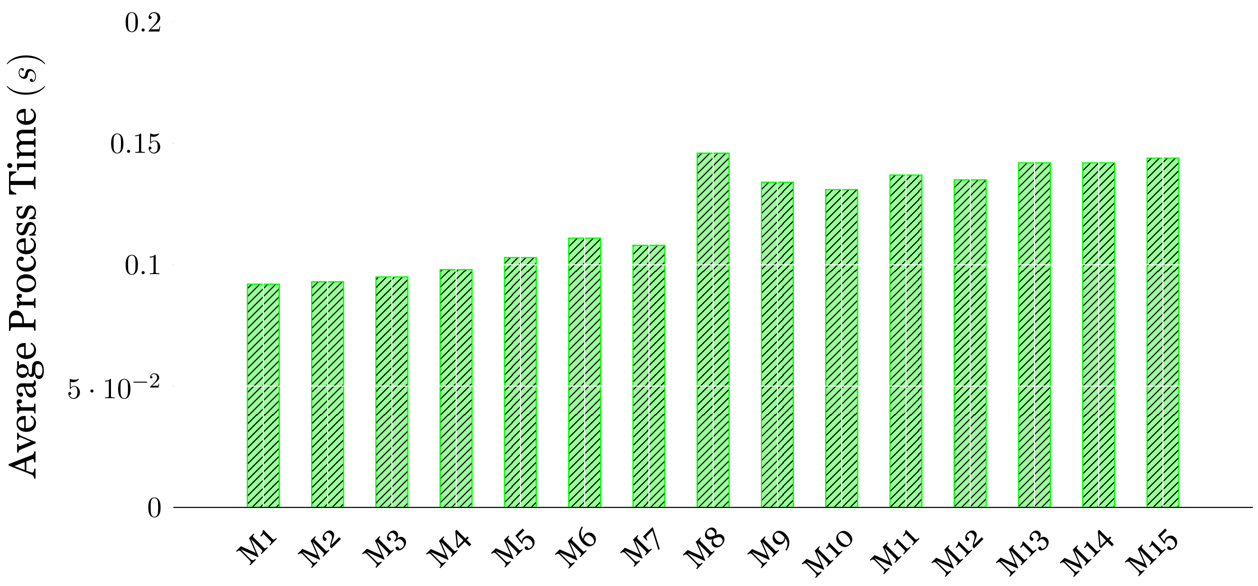

Figure 6.2.1: Average Processing Time per Sample.

As seen from the Table 6.1.1 and Figure 6.2.1, the fusion models which involve early, late, and hybrid fusion take slightly extra time compared to the unimodality-based learner. The late fusion model (M8) has more mean average processing time in comparison to all the other models as it uses five pretrained unimodels which are trained and tested individually,thereby, increasing the time overhead. The early fusion models also show an increase in the average processing time in comparison to the unimodels but lesser than the late fusion model as training the pretrained model after weight enhancement takes more time in comparison to the unimodals. Hybrid learners also show the same trend because of the underlying fact that it is a combination of the early and late fusion methods. These models employ weight enhanced early fusion models adding to the time overhead, besides fusing the scores from different models after validation.

The hyper-parameters values have been fixed while performing the experimental tasks on various models to avoid uncertainties in the comparative study. The learning rate used in all the models is 0.01, with a drop out rate of 0.5. The batch size of the images is fixed to 34, and every model is trained for 1000 epochs with Adam as optimizer to show the consistent comparison across distinctive models.

Chapter 7

Conclusion

This report focuses on the three methodologies to improve the localization accuracy of a pose regressor. The first methodology is a simple unimodal approach utilizing seven distinctive DCNNs. The second methodology chooses the best two unimodal and assists in performing the multimodality-based fusion techniques, namely, early and late fusion. Multimodality-based learners provide a more favorable performance than unimodal learners. The third methodology combines the multimodality-based approaches, thus making a hybrid learner. The above experiments drive an approach to try different DCNNs thereby, giving more opportunities to open for fusion technique. The unimodal with ResNet101 and VGG19 stood ahead of all the existing unimodals. These best performing unimodal helped us to further improve the accuracy by applying fusion technique in terms of the early and late fusion of which the former gives the best result. The hybrid learner incorporating all the best properties of the existing models, shows unusual performance matrices by getting better accuracy i.e., improvement in translation error by a 50% decrease, although the rotation error increase by 4% when compared to unimodal approach using ResNet101 as a feature extractor. This increase in the rotation values accounts for one of the major shortcomings of our experimental analysis. The experiments gradually improved the translation errors but, at the same time, increased rotation errors. While considering a holistic approach, it is found that the hybrid model using averaging is the best performing model as the improvement in the accuracy is drastic.

7.1 Future Work

As the industry for autonomous vehicles is gradually increasing, therefore the need for improvement in its implementation is the principal priority, which includes the minimization of the error so that no machine-made errors would lead to instability of human life as it can cause heavy damage to mankind. So as per our experimental analysis, the drawbacks are quite evident in the increased rotation error values, thus, in a way indicating the first future work to minimize the rotation error. The second future work indicates the improvement in the time overhead. It is visible from the timing analysis that the models involving the multimodal-based learner and hybrid learner consume more time than all unimodal-based learners, thereby making the best performing model less efficient in time analysis.

Bibliography

[1] Durrant-Whyte, H. and Bailey, T. “Simultaneous localization and mapping: part I”. In: *IEEE Robotics Automation Magazine* 13.2 (June 2006), pp. 99–110. issn: 1558-223X. doi: 10.1109/MRA.2006.1638022.

[2] Brubaker, M. A., Geiger, A., and Urtasun, R. “Map-Based Probabilistic Visual Self-Localization”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.4 (Apr. 2016), pp. 652–665. issn: 1939-3539. doi: 10.1109/TPAMI.2015.2453975.

[3] Wang, Peng, Huang, Xinyu, Cheng, Xinjing, Zhou, Dingfu, Geng, Qichuan, and Yang, Ruigang. “The apolloscape open dataset for autonomous driving and its application”. In: *IEEE transactions on pattern analysis and machine intelligence* (2019).

[4] Yamashita, Rikiya, Nishio, Mizuho, Do, Richard Kinh Gian, and Togashi, Kaori. “Convolutional neural networks: an overview and application in radiology”. In: *Insights into imaging* 9.4 (2018), pp. 611–629.

[5] Nwankpa, Chigozie, Ijomah, Winifred, Gachagan, Anthony, and Marshall, Stephen. “Activation functions: Comparison of trends in practice and research for deep learning”. In: *arXiv preprint arXiv:1811.03378* (2018).

[6] Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep learning*. MIT press, 2016.

[7] Taylor, Matthew E and Stone, Peter. “Transfer learning for reinforcement learning domains: A survey”. In: *Journal of Machine Learning Research* 10.Jul (2009), pp. 1633–1685.

[8] Dai, Wenyuan, Yang, Qiang, Xue, Gui-Rong, and Yu, Yong. “Boosting for transfer learning”. In: *Proceedings of the 24th international conference on Machine learning*. 2007, pp. 193–200.

[9] Durrant-Whyte, Hugh and Bailey, Tim. “Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms”. In: *IEEE ROBOTICS AND AUTOMATION MAGAZINE* 2 (2006), p. 2006.

[10] Montemerlo, Michael, Thrun, Sebastian, Koller, Daphne, Wegbreit, Ben, et al. “FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges”. In: *IJCAI*. 2003, pp. 1151–1156.

[11] Montemerlo, Michael, Thrun, Sebastian, Koller, Daphne, Wegbreit, Ben, et al. “FastSLAM: A factored solution to the simultaneous localization and mapping problem”. In: *Aaai/iaai* 593598 (2002).

[12] Smith, Randall C and Cheeseman, Peter. “On the representation and estimation of spatial uncertainty”. In: *The international journal of Robotics Research* 5.4 (1986), pp. 56–68.

[13] Smith, Randall, Self, Matthew, and Cheeseman, Peter. “Estimating uncertain spatial relationships in robotics”. In: *Autonomous robot vehicles*. Springer, 1990, pp. 167–193.

[14] Kendall, Alex, Grimes, Matthew, and Cipolla, Roberto. “Posenet: A convolutional network for real-time 6-dof camera relocalization”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2938–2946.

[15] Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[16] Sundermeyer, Martin, Schlüter, Ralf, and Ney, Hermann. *LSTM Neural Networks for Language Modeling*.

[17] Rumelhart, David E, Hinton, Geoffrey E, and Williams, Ronald J. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), pp. 533–536.

[18] Bengio, Yoshua, Simard, Patrice, and Frasconi, Paolo. “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[19] Walch, Florian, Hazirbas, Caner, Leal-Taixé, Laura, Sattler, Torsten, Hilsenbeck, Sebastian, and Cremers, Daniel. "Image-based Localization with Spatial LSTMs". In: *CoRR* abs/1611.07890 (2016). arXiv: `1611.07890`. url: `http://arxiv.org/abs/1611.07890`.

[20] Napoletano, Paolo, Piccoli, Flavio, and Schettini, Raimondo. "Anomaly detection in nanofibrous materials by CNN-based self-similarity". In: *Sensors* 18.1 (2018), p. 209.

[21] Simonyan, Karen and Zisserman, Andrew. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[22] Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[23] Akilan, T, Wu, QM Jonathan, Safaei, Amin, and Jiang, Wei. "A late fusion approach for harnessing multi-CNN model high-level features". In: *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE. 2017, pp. 566–571.

[24] Akilan, Thangarajah, Wu, QM Jonathan, Yang, Yimin, and Safaei, Amin. "Fusion of transfer learning features and its application in image classification". In: *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE. 2017, pp. 1–5.

[25] Akilan, Thangarajah, Wu, Qingming Jonathan, and Zhang, Hui. "Effect of fusing features from multiple DCNN architectures in image classification". In: *IET Image Processing* 12.7 (2018), pp. 1102–1110.

[26] Xu, Jinwei, Zhao, Yang, Jiang, Jingfei, Dou, Yong, Liu, Zhiqiang, and Chen, Kai. "Fusion Model Based On Convolutional Neural Networks With Two Features For Acoustic Scene Classification". In: *Detection and Classification of Acoustic Scenes and Events 2017* (Nov. 2017).

[27] Fu, Yun, Cao, Liangliang, Guo, Guodong, and Huang, Thomas S. "Multiple feature fusion by subspace learning". In: *Proceedings of the 2008 international conference on Content-based image and video retrieval*. 2008, pp. 127–134.

[28] Bahrampour, Soheil, Nasrabadi, Nasser M, Ray, Asok, and Jenkins, William Kenneth. "Multimodal task-driven dictionary learning for image classification". In: *IEEE transactions on Image Processing* 25.1 (2015), pp. 24–38.

[29] Sun, Yi, Wang, Xiaogang, and Tang, Xiaoou. "Hybrid deep learning for face verification". In: *Proceedings of the IEEE international conference on computer vision*. 2013, pp. 1489–1496.

[30] Pawar, VN and Talbar, SN. "Hybrid machine learning approach for object recognition: Fusion of features and decisions". In: *Machine Graphics and Vision* 19.4 (2010), pp. 411–428.

[31] Kendall, Alex and Cipolla, Roberto. "Geometric loss functions for camera pose regression with deep learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5974–5983.

[32] Botchkarev, Alexei. "Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology". In: *arXiv preprint arXiv:1809.03006* (2018).

[33] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[34] Akilan, T., Wu, Q. M. J., Safaei, A., and Jiang, W. "A late fusion approach for harnessing multi-cnn model high-level features". In: *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. Oct. 2017, pp. 566–571.

[35] Akilan, T., Wu, Q. M. J., Yang, Y., and Safaei, A. "Fusion of transfer learning features and its application in image classification". In: *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*. Apr. 2017, pp. 1–5.

[36] Kuang, Hongwu, Liu, Xiaodong, Zhang, Jingwei, and Fang, Zicheng. "Multi-Modality Cascaded Fusion Technology for Autonomous Driving". In: *arXiv preprint arXiv:2002.03138* (2020).

[37] Bouthellier, Paul. "ROTATIONS AND ORIENTATIONS IN R3". In: *27th International Conference on Technology in Collegiate Mathematics* 27 (2015).

[38] Kabir, Waziha, Ahmad, M Omair, and Swamy, MNS. "Weighted hybrid fusion for multimodal biometric recognition system". In: *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. 2018, pp. 1–4.

[39] Sakai, Takao, Kimura, Daiki, Yoshida, Taichi, and Iwahashi, Masahiro. “Hybrid method for multi-exposure image fusion based on weighted mean and sparse representation”. In: *2015 23rd European Signal Processing Conference (EUSIPCO)*. IEEE. 2015, pp. 809–813.

Appendix

Appendix A

IEEE Permission to Reprint

Appendix B

Gantt chart

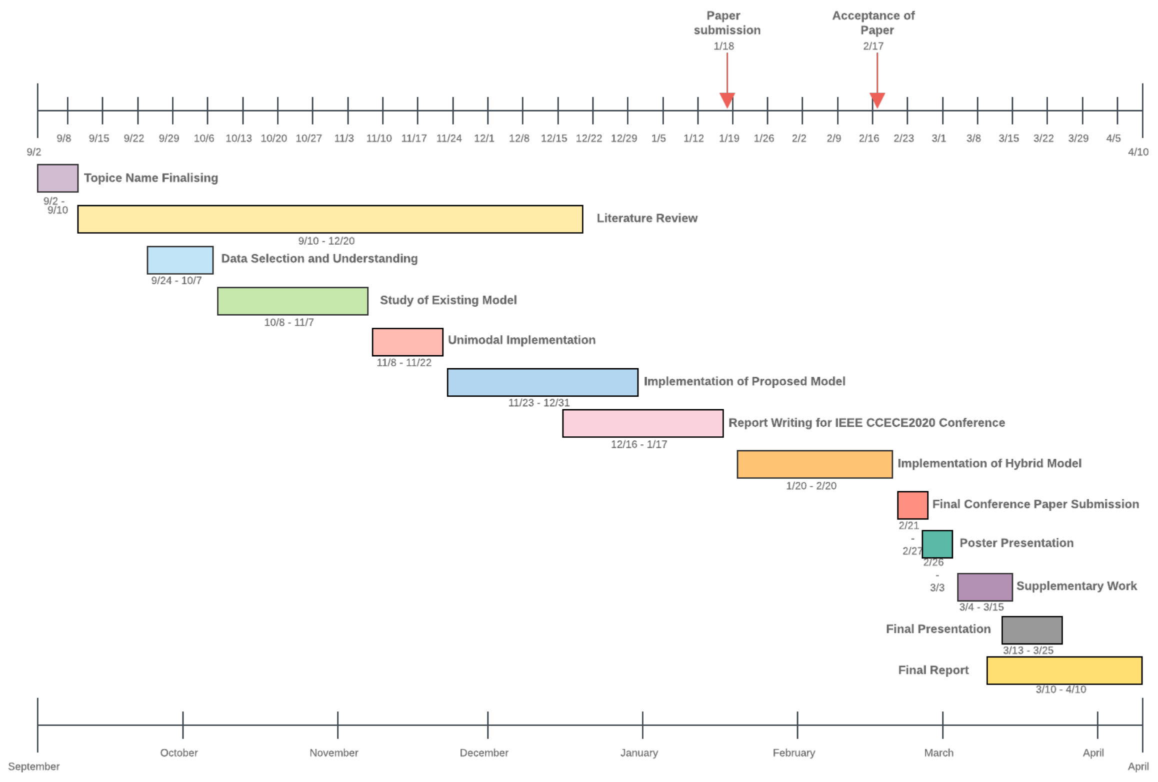

Figure B.0.1: Timeline Chart for the Project

Appendix C

Supplementary developments

Model Name	Median e_t (m)	Mean e_t (m)	Median e_r ($^\circ$)	Mean e_r ($^\circ$)
M_1	17.015	36.018	0.836	6.273
M_2	11.963	85.477	1.088	9.563
M_3	9.522	13.516	1.133	4.646
M_4	20.178	23.521	1.279	3.653
M_5	17.319	19.787	1.446	2.492
M_6	19.404	22.225	1.600	2.897
M_7	10.713	12.835	1.308	2.965
M_8	14.708	17.592	1.508	3.520
M_9	10.523	12.441	0.886	3.934

Table C.0.1: Performance Analysis of Translation (e_t) and Rotation (e_r) errors after supplementary developments.

The table above shows the extended development work besides the one documented in this report. The results of all these experiments are not satisfactory. Therefore, these are rejected models which were not included in the report. M_1 depicts the clustering with bigger cluster using KNN on the original late fusion model L_f. M_2 is the clustering with the smaller cluster using KNN on L_f. M_3 represents the removal of the model that has the farthest pose coordinates from the ground truth. The farthest model was chosen based on the delta parameter which was chosen to be the distance between the pose coordinates and the ground truth. M_4 is the model chosen on the basis of the delta parameter which is the median of the predicted poses. M_5 represents the model chosen on the basis of the delta parameter as the minimum of the predicted poses. M_6 similarly, is the model chosen as the maximum of the predicted poses. M_7, M_8, and M_9 are hybrid fusion models using min, max and median arithmetic operations respectively.

Appendix D

GitHub repository

https://github.com/gauravt1996/Visual-Simultaneous-Localization-and-Mapping