Efficient Image Based Localization Using Machine Learning Techniques

by

Ahmed Elmougi

A Dissertation Submitted in Partial Fulfillment of the

Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Electrical and Computer Engineering

Efficient Image Based Localization Using Machine Learning Techniques

by

Ahmed Elmougi

Supervisory Committee

Dr. Xiaodai Dong, Supervisor

(Department of Electrical and Computer Engineering)

Dr. T. Aaron Gulliver, Department Member

(Department of Electrical and Computer Engineering)

Dr. Yvonne Coady, Outside Member

(Department of Computer Science)

ABSTRACT

Localization is critical for self awareness of any autonomous system and is an important part of the autonomous system stack which consists of many phases including sensing, perceiving, planning and control. In the sensing phase, data from on board sensors are collected, preprocessed and passed to the next phase. The perceiving phase is responsible for self awareness or localization and situational awareness which includes multi-objects detection and scene understanding. After the autonomous system is aware of where it is and what is around it, it can use this knowledge to plan for the path it can take and send control commands to pursue this path. In this proposal, we focus on the localization part of the autonomous stack using camera images. We deal with the localization problem from different perspectives including single images and videos.

Starting with the single image pose estimation, our approach is to propose systems that not only have good localization accuracy, but also have low space and time complexity. Firstly, we propose SurfCNN, a low cost indoor localization system that uses SURF descriptors instead of the original images to reduce the complexity of training convolutional neural networks (CNN) for indoor localization application. Given a single input image, the strongest SURF features descriptors are used as input to 5 convolutional layers to find its absolute position and orientation in arbitrary reference frame. The proposed system achieves comparable performance to the state of the art using only 300 features without the need for using the full image or complex neural networks architectures. Following, we propose SURF-LSTM, an extension to the idea of using SURF descriptors instead the original images. However, instead of CNN used in SurfCNN, we use long short term memory (LSTM) network which is one type of recurrent neural networks (RNN) to extract the sequential relation between SURF descriptors. Using SURF-LSTM, We only need 50 features to reach

comparable or better results compared with SurfCNN that needs 300 features and other works that use full images with large neural networks.

In the following research phase, instead of using SURF descriptors as image features to reduce the training complexity, we study the effect of using features extracted from other CNN models that were pretrained on other image tasks like image classification without further training and fine tuning. To learn the pose from pretrained features, graph neural networks (GNN) are adopted to solve the single image localization problem (Pose-GNN) by using these features representations either as features of nodes in a graph (image as a node) or converted into a graph (image as a graph). The proposed models outperform the state of the art methods on indoor localization dataset and have comparable performance for outdoor scenes.

In the final stage of single image pose estimation research, we study if we can achieve good localization results without the need for training complex neural network. We propose (Linear-PoseNet) by which we can achieve similar results to the other methods based on neural networks with training a single linear regression layer on image features from pretrained ResNet50 in less than one second on CPU. Moreover, for outdoor scenes, we propose (Dense-PoseNet) that have only 3 fully connected layers trained on few minutes that reach comparable performance to other complex methods.

The second localization perspective is to find the relative poses between images in a video instead of absolute poses. We extend the idea used in SurfCNN and SURF-LSTM systems and use SURF descriptors as feature representation of the images in the video. Two systems are proposed to find the relative poses between images in the video using 3D-CNN and 2DCNN-RNN. We show that using 3D-CNN is better than using the combination of CNN-RNN for relative pose estimation.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

my supervisor, Dr. Xiaodai Dong, for the opportunity she gave to start my study under her supervision, her mentoring since the beginning, and her patience even my progress was slow in the beginning and her trust and advice guided me to all the achievements done during my study. I will be always in favour for all the things she helped me with and grateful to be her student.

Dr. Tao Lu for all the helpful advice he gave to me and all the time he spent to guide me through my PhD.

Dr. T. Aaron Gulliver, and Dr. Yvonne Coady, for all the helpful comments that helped modifying the thesis and finishing it in the best possible shape.

Dr. Hoang Minh Tu, He has been like a brother in all the years, helped me since the first day, he is always supportive and sheer for my success and provide all kinds of help,Thank you.

My wife, Zeinab Elashry, for all the love and care she gave me during this hard journey, for all the sacrifices she made to make this happen, she always accepted me regardless of anything, always believe in me that I will succeed from the first day despite that the road was not clear, Thank you for everything.

My family, for all the support and the prayers and all the efforts to make sure I receive a proper education. I will always try to make you prouder and will always be in favour for all you did.

Ahmed Elmoogy, Victoria, BC, Canada

DEDICATION

To all who are starting from scratch to learn anything, this is an example that everything is possible and you can do it.

Chapter 1

Introduction

1.1 Motivations

Any autonomous system is required to localize itself at any environment using the equipped sensors. It can be a self driving car travelling across the streets or an autonomous robot providing services for customers at indoor environments such as restaurants or shopping malls [15]. Localization algorithms differ depending on the type of the environments and the sensors used for localization. Outdoor scenes are very complex while indoor scenes are usually simpler. Moreover, many sensors data are available to use for localization systems including cameras images , light detection and ranging (LiDAR) and radio detection and ranging (RADAR) scans and wireless signals such as wireless fidelity (WiFi) signals. One of the widely used sensors for perception is camera on which some self driving cars companies including Tesla depend fully for the achieving full autonomy due to its low cost compared to LiDARs and high visual information data which we can analyze using the existing AI systems [16]. In this thesis, we focus on using camera images for localization.

The proposed systems lies into two categories:

1. Single image absolute pose estimation

2. Multiple images (video) relative pose estimation

The single image absolute pose estimation systems try to learn where this image was taken in any environment which makes these systems hard to generalize to different environments. Therefore, the main objective of the proposed systems is not only to provide good localization accuracy for both indoor and outdoor scenes, but also implement them in the most efficient way in terms of storage space, training and testing time of the machine learning algorithms in order to make fine-tuning these systems more efficient. One of the most important applications of single image pose estimation is for re-localization or kidnapped robot problem [1] where the agent is moving in any environment and lost tracking of where it is right now or the visual odomotery system used has a big drift and we need to correct. In this circumstance, the agent can take one image of its current view and use it with the single image pose estimation systems to know where it is now and then we can initialize the visual odometery system afterwards.

Multiple images are used instead of single image systems to find the relative poses to make the system generalizes to unknown scenes as the system does not depend on the scale of the environment. We go through the proposed research issues in the following sections.

1.2 Research Objectives and Contributions

1.2.1 SurfCNN: A Descriptor Accelerated Convolutional Neural Network for Image-based Indoor Localization

In Chapter 2, we propose SurfCNN, a low complexity image-based indoor localization system using convolutional neural network (CNN). Given a single image, the CNN learns to predict the absolute pose (position and orientation) of this image in an arbitrary reference frame. However, instead of using the full image as input to a complex CNN, we propose extracting speeded up robust features (SURF) features [17] first along with their descriptors that describe the area around every extracted feature and use these SURF descriptors as input to small CNN architecture. As a result, the input size is reduced by a factor of 48 or more with training in minutes instead of hours needed by the other single image localization systems and reaching comparable performance. The proposed research issues are identified as follows:

1. The analysis the single image localization problem.
2. The detailed structure of proposed system.
3. The time analysis for SurfCNN compared to the other works.
4. The localization accuracy in terms of the median position and orientation errors for 3 indoor datasets.

1.2.2 SURF-LSTM: A Descriptor Enhanced Recurrent Neural Network For Indoor Localization

Chapter 3 extends the idea of using SURF descriptors instead of the original images for single image indoor localization. However, instead of using CNN to extract the

features from the images' descriptors, we propose SURF-long short memory (SURF-LSTM) that uses recurrent neural networks to extract the sequential features between the strongest SURF features descriptors. Bidirectional LSTM (Bi-LSTM) [18] network is used to extract the features from the input images features. Using SURF-LSTM, comparable localization accuracy is achieved using lower number of features than SurfCNN with a huge reduction in both the training time and the input dimension compared to the other works. The proposed research issues are identified as follows:

1. The analysis of the SURF-LSTM structure and design consideration.
2. The storage space and time analysis of the proposed system.
3. The localization performance analysis using two indoor datasets.

1.2.3 Pose-GNN : Camera Pose Estimation System Using Graph Neural Network

In Chapter 4, We formulate the single image pose problem in the form of a graph learning problem and we use graph neural network (GNN) to learn the image pose estimation. We propose two GNN architectures by either representing the image as a node in a big graph or characterizing the image itself as a graph using the image's features extracted from pre-trained CNN architecture. For both systems, we use K-nearest neighbours algorithm [19] to assign neighbours for each node. The proposed systems outperform all the other methods for the indoor scenes and have competitive performance for the outdoor scenes. The research perspectives are as follows:

1. The details of the proposed architectures.
2. The effect of the number of neighbours on the performance.

3. The validation of the proposed systems using two indoor and outdoor scenes.

1.2.4 Efficient Camera Pose Estimation Using Linear Regression and PCA

Chapter 5 discusses the following question, do we need to train very complex neural networks systems to reach good accuracy of localization?. We show that using only one layer of linear regression on top of the features of pretrained CNN, we can reach comparable results to the neural networks-based single image localization systems for indoor scenes with training time less than a second and without the need for GPU. Moreover, for outdoor scenes, we propose 3 fully connected layers architecture that can use pretrained features as they are without fine-tuning and reach comparable or better results to the state of the art. We also show that downsampling the input features using principal components analysis (PCA) does not severely degrade the performance but can make the training time faster. The proposed research issues are identified as follows:

1. The discussion of the proposed architectures.

2. The hyper-parameters tuning and design analysis.

3. The validation of the proposed systems using two indoor and outdoor scenes.

1.2.5 Generalizable Sequential Camera Pose Learning Using Surf Enhanced 3D CNN

In Chapter 6, we use SURF descriptors for video localization instead of single image localization presented in Chapters 2 and 3. As the single image pose estimation systems require re-training on every new environment, the main goal of the proposed

work is to make the system generalize to scenes that are different to the training scenes. We propose two neural networks architecture based on CNN-RNN and 3D-CNN to find the relative poses between images in a video of an arbitrary number of frames. The proposed systems are able to generalize well to unknown scenes while achieving competitive performance to the state of the art. The proposed research issues are identified as follows:

1. The comparison between 3D-CNN and CNN-RNN architectures.
2. The generalization capability analysis.
3. The localization accuracy analysis using an indoor dataset.

1.3 Hardware specification

All the experiments in this dissertation are done using a work station with the following specifications:

- Intel Xeon CPU with 8 cores and 2 GHZ CPU clock rate
- DIMM RAM with 1333 MHz speed and 55 GB memory size
- Nvidia Titan XP GPU with 3840 Nividia CUDA Cores, 1582 MHZ boost clock, 11.4 Gbps memory speed, 12 GB GDDR5X memory size and 547.7 GB/s memory bandwidth

Chapter 2

SurfCNN: A Descriptor Accelerated Convolutional Neural Network for Image-based Indoor Localization

2.1 Introduction

Internet of things (IoT) has found widespread use in different industries by integrating computing, communications and control functions into a network formed by sensor nodes, edge devices and remote servers. The raw data collected by sensors are eventually sent to cloud servers with or without processing at the sensor and edge nodes. The distribution of computing to IoT network components is tailored based on the capacity of communications links, computing resources at each level, and the delay requirement of the applications. As more intelligence is being built into IoT, neural networks play an important role in data processing. Among various deep learning

tools, convolutional neural networks (CNN) are capable of extracting features from high dimensional data such as images, videos, etc., that suits the application specific tasks. Such neural network, however, requires high dimensional optimization procedure in which the training time is significantly longer when the input dimension is large. This poses significant challenges to communications links to transport a large amount of raw data to a cloud server for CNN training and inference. If inference is shifted to edge and/or sensor nodes, the computing resources are usually very limited in comparison to the CNN computational complexity. Therefore, ways to reduce the CNN dimension and complexity are necessary to facilitate the use of CNN in IoT applications. It is well known that image descriptors extract features from images through deterministic means that are orders of magnitude faster than CNN. The drawback of a descriptor is that, the output feature size is usually large compared to those from CNN as most of the image information is retained during extraction regardless of whether it is needed for the target application. Inspired by the characteristics of CNN and image descriptors, here, through the demonstration from an indoor localization application, we combine both technologies by first using an image descriptor to extract features from images. The feature set, which has significantly reduced dimension compared to the images, is input to a CNN to extract more useful features with further reduced dimension. The combined techniques result in a significant fewer parameters in the CNN and reduced training and testing time, which substantially lower the required communications and computing resources and the overall latency.

People normally spend over 87% of their daily lives indoors [20]. Consequently, indoor localization is important to many applications ranging from indoor navigation to virtual reality [21]. Unlike outdoor localization where the global positioning system (GPS) is prevalent, indoor localization relies on readings from sensors such

as received signal strength indicator (RSSI), light detection and ranging (LIDAR), inertial measurement unit (IMU) and images from cameras as GPS signals are too weak [22]. In particular, image based localization is popular in many fields including robotics [23] and simultaneous localization and mapping (SLAM) [24] for its high accuracy and low cost.

Image based localization methods can be categorized into two sub-categories: feature based and direct methods [25]. In direct methods, all the image pixels are used for location estimation process, making it computationally expensive despite of its capability to generating dense maps [26]. On the other hand, feature based methods extract features from the images and estimate the location using these features. Consequently it is much faster than direct methods. For feature based methods, there have been various feature detectors for feature extraction, including scale-invariant feature transform (SIFT) [27], sped up robust feature (SURF) [17], features from accelerated segment test (FAST) [28], binary robust independent elementary features (BRIEF) [29] and oriented FAST and rotated BRIEF (ORB) [30], etc. A typical procedure for feature based localization is as follows: features are first extracted using one of these algorithms. Then, descriptors are estimated to identify the area around each feature for matching. Following, either the consecutive image descriptors are matched to find the relative pose (position and orientation) between them [31], or each image descriptors are matched against a large database of key frames or landmarks with known locations to identify what is the closest match to infer the current pose of the camera [32].

Although good performances have been achieved by these classical methods, they have drawbacks. For example, the accuracy of all these methods rely on a good estimation of initial pose location. In addition, the pose accuracy is determined by the matching between descriptors. Therefore, wrong correspondences can accumulate

errors. Finally, large database is required for image retrieval methods with a size proportional to the area of the scene.

Recently, convolutional neural network (CNN) [33], which is capable of extracting features from high dimensional data including images, videos, etc., has been a popular choice for indoor localization. Multiple CNN architectures including AlexNet [34], GoogleNet [35] and ResNet [36] achieve state of the art performance in multiple applications. For image based indoor localization, it is typically implemented by training an end to end architecture to learn the global pose of a single image with respect to a known reference frame [1]. However, CNN requires high dimensional optimization procedure in which the training time is long when the input size is large. Furthermore, CNN need to be re-trained or fine tuned when the testing scene is significantly different from the training scene.

To reduce the complexity of CNN, we propose, for the first time according to the authors' knowledge, to use the image features' descriptors instead of the image itself as input to CNN to learn the global image pose. Among all features descriptors and detectors, SURF descriptors are used extensively in computer vision applications such as face recognition [37], visual simultaneous localization and mapping (SLAM) [38] and object detection [39]. In addition, [40] demonstrates that the SURF descriptor reaches the highest accuracy in image matching for indoor localization. Typically, SURF can convert an image with around 1 million pixels into feature set with fewer than 20 thousand values by taking the strongest 300 features' descriptors, sorted by the corresponding Hessian threshold. This significantly reduces the data dimension without noticeable loss of image information. The proposed method aims to combine the advantages of both the classical methods and the direct CNN method while mitigating their disadvantages. Our system does not require an initial pose and the correspondences between descriptors or a large database. Moreover, it reduces the

complexity of the direct CNN by reducing the input dimensions to CNN for model simplification, easier training and re-training on different environments, and faster inference. This greatly facilitates its use in edge and cloud computing with less demanding data storage, transmission and computation requirements.

2.2 Related Work

2.2.1 Descriptors Related CNN Model

CNN has been used to extract the descriptors of an input image which is either a full image or a pair of non corresponding patches of the image. For example, Siamese CNN is proposed in [41] to learn the descriptors of small patches of images to achieve patch matching at 95% recall rate. The same approach is employed in [42] but using pairs of non corresponding patches where the network learns 128-D descriptors whose Euclidean distances reflect patch similarity. [43] uses similar idea of using pair of patches to learn a similarity score from. Further, unsupervised learning [44] with VGG-16 network architecture [45] is introduced [46] to learn compact binary descriptor for efficient visual object matching.

2.2.2 Image-based localization

Using images for localization is a part of the visual SLAM [25]. In the past, most classical visual SLAM systems used either handcrafted local features extracted (feature-based methods) or the whole image without extracting any features (direct methods). In feature-based methods, [31, 47] employ ORB descriptor matching to find the correspondences between consecutive frames and then uses the 8-point algorithm [48] with bundle adjustment [49] to find the relative pose between frames. Other techniques [50, 51] rely on finding the 2D-3D correspondence via descriptor matching

between the 2D image and 3D model using e.g., structure from motion (SfM) [52].

Paper [26] is a direct method that uses the full image information. It provides a denser map but without affecting the pose estimation significantly compared to [31]. In addition, [53] uses image retrieval techniques to search for the similarity between the current image and images in the database. Consequently, the position at which current image was taken can be estimated.

With the boom of deep learning, neural networks are used in visual SLAM and image based localization by mapping directly from single image space to absolute position while circumventing challenges in classical methods. For example, transfer learning [54] and the pretrained GoogLeNet [55] are employed in [1] to regress the pose of the camera. In later publications, the same authors improve the accuracy by modifying the loss function according to the geometry, the relation between position and orientation [11], or the uncertainty calculation [5]. Further, [4] generates uncertainty for the camera poses using Gaussian process regressors. Several orientation representations and data augmentation are introduced in [6] along with a complex networks with shared convolutions to learn the position and orientation. Recently, [2] adopts long-short term memory (LSTM) [56] to similar model to memorize good features. In [36], pretrained ResNet-34 is applied for regressing camera pose. It adopts encoder-decoder design and uses skipped connection to move the features from the early layers to the output layers.

Despite the outstanding performance of the neural networks-based systems, the training of these networks is time consuming. The complexity of training increases if the model needs to be fine tuned when the testing images are taken in completely different environments from what the training images taken from. To reduce the training complexity, we propose a novel technique to reduce the input dimensions and consequently the parameter size of the CNN.

2.3 Network Model

In this section, we apply our model to image based indoor localization application. The task is that given an image I taken by a camera with unknown intrinsic parameters with corresponding SURF descriptors vector D, the network learns the camera pose in the form of global Cartesian position $[x, y, z]^T$ and orientation in quaternion form $[q_w, q_x, q_y, q_z]^T$. Note, in contrast to the rotation matrix and Euler angles, quaternion is a stable form to represent orientation. Consequently, both position and orientation can be combined to form the output pose vector $\boldsymbol{P}$ $[x, y, z, q_w, q_x, q_y, q_z]^T$ of size 7 of the camera pose.

2.3.1 Dataset

Since our system is designed for working in indoor environments, the training and testing are performed on 3 well known indoor datasets:

1. Microsoft RGB-D 7 scenes [57] dataset: It is composed of 7 scenes with constantly changing views and varying camera heights as seen in Fig. 2.2. It contains both the RGB image and depth map with corresponding pose.

2. RGB-D SLAM Dataset and Benchmark [58]: It is a benchmark for the evaluation of visual odometry and visual SLAM systems. It contains RGB and depth images with the ground truth trajectory. We choose to work with 4 scenes, each of which provides sequences for training with ground truth poses and others for testing without ground truth positions. An online tool is available for evaluation of the testing scenes.

3. The ICL-NUIM dataset [59]: It is a recently developed benchmark similar to the RGB-D SLAM benchmark dataset for RGB-D, visual odometry and SLAM al-

gorithms. We will work on the office room scene with 4 sequences, one sequence for training and the other 3 for testing.

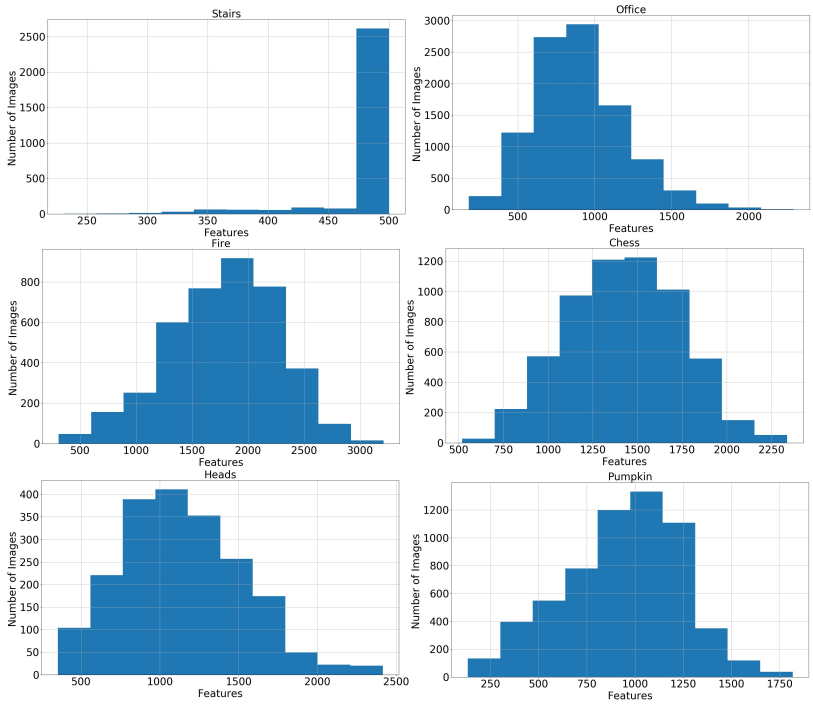


Figure 2.1: The histogram of features for Stairs, Office, Fire, Chess, Heads and Pumpkin scenes.

2.3.2 SURF Descriptors

The SURF algorithm [17] starts with computing the interest points or features. To find the correspondences between these features, the description of the area around these features is computed for robust matching. This is called the descriptor. In the original SURF implementation, there are two ways to find the descriptor of the detected feature depending on its rotational invariance. To make the descriptors invariant to rotations, the dominant orientation of a neighbourhood around the feature is calculated using Haar-wavelet transform in both x and y direction. Subsequently, the area around the feature is divided into 4×4 grid along the dominant orientation.

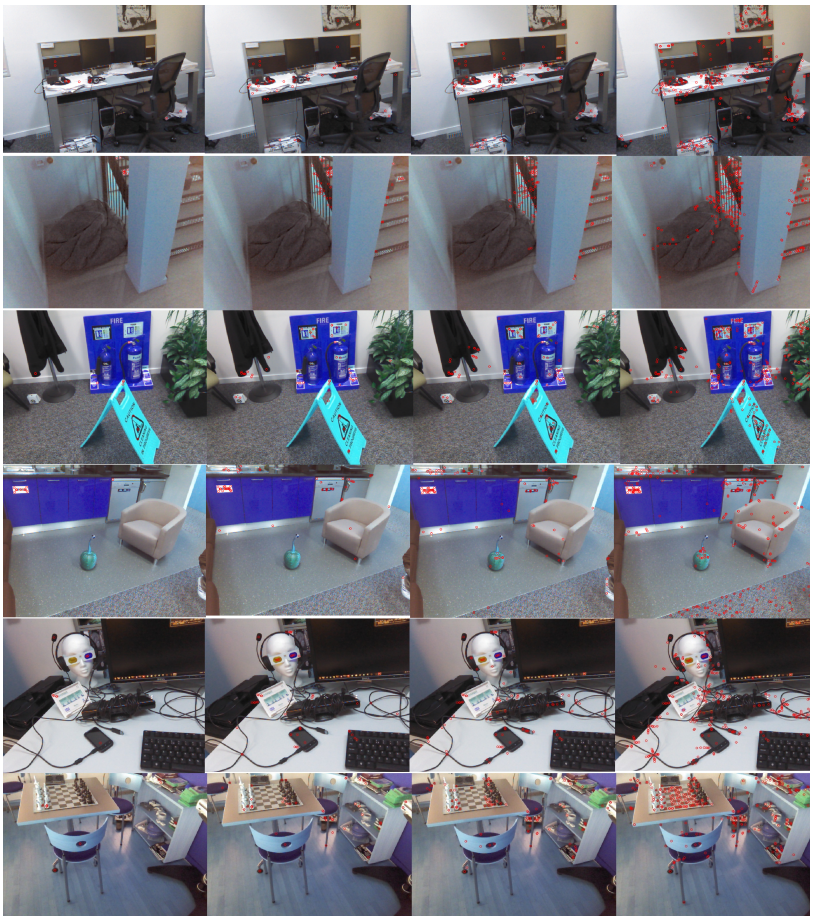


Figure 2.2: Illustrations of image feature extraction using SURF descriptor. Red dots represents the location of (from left to right:) 10, 50, 100 and 300 SURF features of (from top to bottom:) office, stairs, fire, pumpkin, heads and chess scenes.

As opposed to the computationally expensive counterpart SIFT [27], Haar-wavelet transform [60] is obtained for both vertical and horizontal directions. The absolute values of all vertical and horizontal responses are summed to form the descriptor vector of all the sub-regions around the feature with size 64. In the case where the orientation information needs to be maintained within the SURF descriptor, the dominant orientation calculation step is skipped and the resulting vector is called Upright SURF (U-SURF) descriptor. In our models, we use both SURF and U-SURF descriptors and demonstrate that using U-SURF does enhance the orientation calculation compared to the ordinary SURF descriptor.

Fig. 2.1 displays the histogram of features for images of the 7 scenes dataset. As shown, the number of features ranges from 100 to 4000. To make the input dimension feasible, we choose to work with the full 300 features or fewer. Here, each feature has 64 values and the maximum input vector size is 300×64. Compared to the origin image of 480×640×3 pixels, use of the discriptor reduces the CNN input size by at least a factor of 48.

Further, the location of the 300, 100, 50 and 10 most important SURF features are shown in Fig. 2.2 as red dots. It is seen that the SURF algorithm focuses on feature points such as edges and corners, making them good representative of images that uniquely describe every feature's surrounding area. Consequently, the majority of the information in the image is retained in SURF descriptors.

2.3.3 The Loss Function

We choose to represent the output pose as one 7 dimensional vector instead of having two outputs for position and orientation in [1]. Having one output vector makes it easy to train using one loss function instead of having two parts of the loss function with a weight factor between them that needs to be tuned for every scene. Consequently, the objective function is

$$loss(D) = \left\| \hat{P} - P \right\|_2 \tag{2.1}$$

where D is the image descriptors vector, $\hat{P}$ is the predicted pose, and P is the ground truth pose.

2.3.4 Architecture

Our model is composed of as few as 7 layers as shown in Fig. 2.3. Here, 5 convolutional layers with max pooling layers are adopted to reduce the dimensions of the

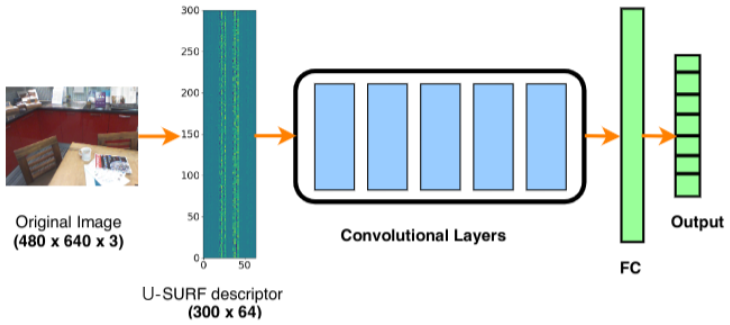


Figure 2.3: The architecture of Surf-CNN

layers output, RELU activation function and batch normalization [61], resulting in faster learning. The first convolutional layer has strides of (2,2), and the remaining 4 layers have (1,1). They are followed by a fully connected (FC) layer of 2,048 neurons with ReLU activation function. To prevent over-fitting, dropout with probability of 0.5 [62] is adopted. The output layer has 7 neurons with linear activation for the pose $[x, y, z, q_w, q_x, q_y, q_z]^T$. The details of the architecture is shown in Table 2.1. The hyper-parameter tuning procedure to arrive at the architecture is summarized as follows. We begin the first layer with filter size 7×7 and then decrease it subsequently to 5×5 and 3×3 as the dimension of the feature map decreases along the network. The number of filters in the first layer is 64 and then increased to 128 and 256 in the subsequent layers following the well known CNN architectures [63]. The number of layers is chosen based on experiments on the positional error. Moreover, the trade off between the number of convolutional layers and the positional error of the heads scene of the 7 scenes dataset is shown in Fig. 2.4. As shown, increasing the number of convolutional layers up to 5 layers leads to a reduction of the error to 0.17 m; however, the error starts to increase with the addition of more layers to 0.65 m with 10 convolutional layers. One reason for the error increase is that every convolutional layer has a pooling layer which downsamples the feature maps, so with 10 convolutional layers, the input shrinks in size to a level where the important features are lost.

Moreover, using SURF descriptors with low dimensions helps reduce the number of layers compared to other CNN-based algorithms that use pretrained networks as the descriptors represent the extracted features from the image which makes the learning process easier than using the raw images.

Table 2.1: Dimensions of CNN layers where N is the number of SURF features

layer	**Dimensions**
Input	$(N, 64, 1)$
conv 1	(64,7,7)
conv 2	(128,5,5)
conv 3	(256,5,5)
conv 4	(256,5,5)
conv 5	(256,3,3)
FC 1	2048
Output	7

Unlike the previous works [1, 4, 5], our system does not require any pre-processing such as cropping or subtracting the mean for the images. The extraction of the SURF descriptors shown in Fig. 2.3 only takes 0.9 seconds per image on average and can be done off-line or in real time.

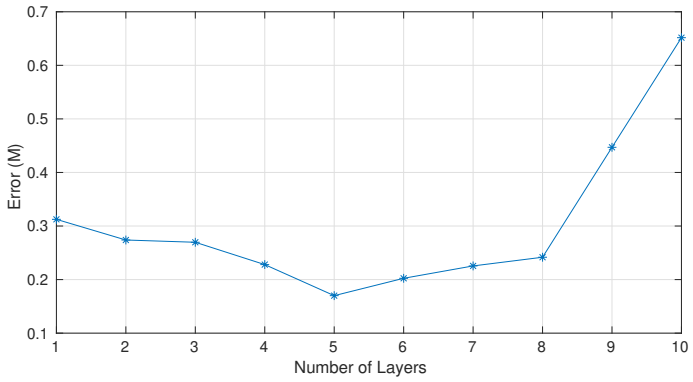


Figure 2.4: The effect of increasing the convolutional layers on the positional error for Heads scene of the 7 scenes dataset.

2.4 Performance Analysis

To compare the SURF and the U-SURF descriptors, we plot the position and orientation errors for the 7 scenes dataset as bar plot in Fig. 2.5. As shown, the U-SURF descriptors, having the orientation information of the features detected, have more precise orientation estimates than using the SURF descriptors with an average error margin of 6°. In addition, using the U-SURF enhances the position estimation, not only with the same extent as orientation but also with an average error difference of 4.5 cm. Evidently, using the U-SURF descriptors is better than using the SURF descriptors for the localization application where the orientation of the features plays an important role. As U-SURF descriptors are rotationally variant, every rotation in the image will give different descriptors depending on the orientation of the area around the feature which will benefit localizing the image and especially finding the orientation of the image as shown on Fig. 2.5.

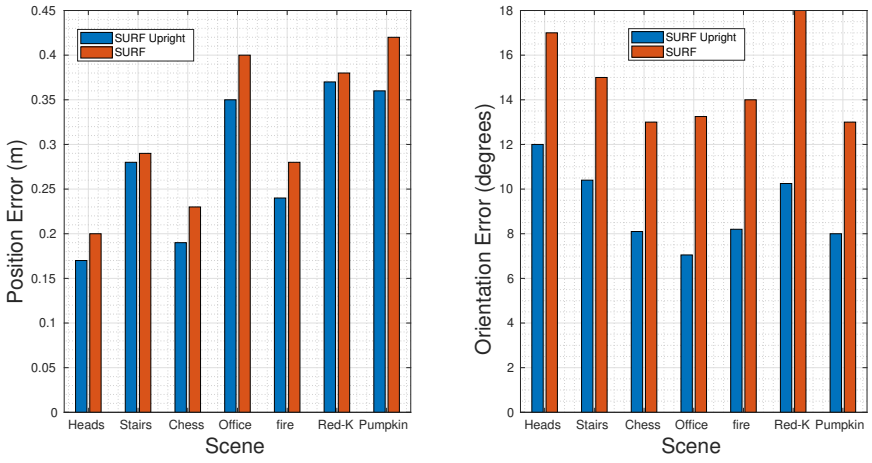


Figure 2.5: The bar plot for the position error (left) and orientation error (right) of U-SURF (in brown) and SURF (in blue) for the 7 scenes dataset.

The main advantage of our SURF accelerated CNN (SurfCNN) is the increase of the speed of training and testing and the reduction in the memory requirements for

storing both the input data and network parameters, which accordingly lowers the amount of data for transmission in edge computing and cloud computing applications. Regarding input data we reduce the input image dimensions from 480×640×3 to descriptor vectors with sizes ranging from 300×64 to 1×64 with a reduction factor ranging from 48 to 14,400. This substantial reduction is critical for low memory systems as the SURF descriptors extraction process can be done efficiently. Further, Table 2.2 shows the number of layers and number of learning parameters of our SurfCNN with features size ranging from 1 to 300 in comparison with the previous work. As shown, SurfCNN reduces the number of layers to 7 layers compared to 24 or more layers in the previous work. In particular, a reduction factor of 1.6 or more is achieved for the number of learning parameters when all 300 features are adopted and 6.2 or more when as few as 1 feature is used. Furthermore, in previous works, pre-trained network may take weeks to train while our model does not require to be pre-trained.

Table 2.2: The number of layers for the current work SurfCNN with 300x64 input features, the average number of parameters and the median error in position, compared with PoseNet [1], Pose-LSTM [2] and Pose-Hourglass [3]

Network	**Layers**	**Pretrained Network**	**Pretrained Parameters**	**Total Parameters**
SurfCNN 300	7	None	0	1.31×10^7
SurfCNN 100	7	None	0	6.61×10^6
SurfCNN 1	7	None	0	3.47×10^6
PoseNet	24	GoogLeNet	1.10×10^7	2.35×10^7
Pose-LSTM	28	GoogLeNet	1.10×10^7	2.15×10^7
Pose-Hourglass	35	ResNet-34	2.30×10^7	4.50×10^7

Fig. 2.6 shows the trade-off between training time and average position error for the 7 scenes dataset with various number of features for our SurfCNN. It illustrates that the relation between the number of features and training time is nearly linear with a minimum time of 15 minutes for using 1 features and max time of 90 minutes for 300 features. Additionally, the inverse relationship between the average error and training time is shown where a maximum average error of 0.41 m with only 15

minutes training time and 0.28 m for 90 minutes are obtained. The training was done using NVidia TITAN XP GPU and Adam solver [64] with a learning rate of 0.0001. Using this visualization, we can choose the number of features needed for SurfCNN according to the target training time, the computational resource availability and maximum allowable error. Overall, SurfCNN is superior to other work that use the full images as input.

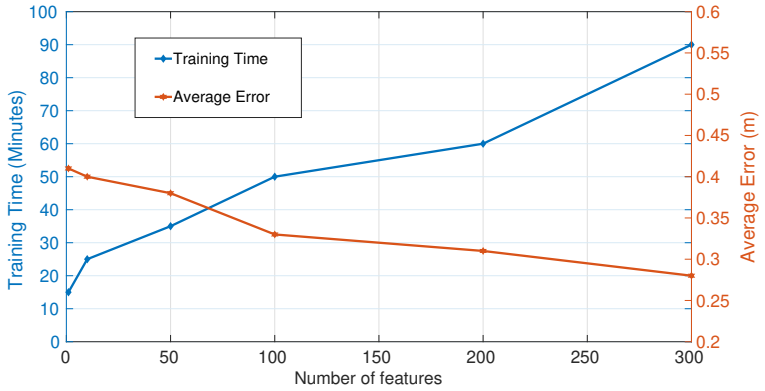


Figure 2.6: The training time (in blue) and the average error (in orange) versus the number of features.

Table 2.3: The median error in position (m)/ orientation (degrees) for the 7 scenes, with 5 input sizes 300×64, 100×64, 50×64, 10×64 and 1×64, compared with PoseNet [1], G-Posenet [4], Posenet-U [5], Pose-L [2], BranchNet [6], Pose-H [3], VidLoc [7], RelocNet [8] and Mobile-PoseNet [9].

Algorithm	**Chess**	**Office**	**Pumpkin**	**Kitchen**	**Stairs**	**Heads**	**Fire**	**Average**
SurfCNN 300	0.19/8.10	0.35/7.05	0.36/10.80	0.37/10.25	0.28/10.14	0.17/12	0.24/8.20	0.28/9.17
SurfCNN 100	0.23/10.20	0.38/9.50	0.38/9.52	0.45/12.35	0.32/11.25	0.19/12.50	0.26/9.25	0.31/10.65
SurfCNN 50	0.26/10.50	0.42/10.31	0.39/13.20	0.47/13.20	0.34/12.22	0.21/13	0.28/10.52	0.33/11.85
SurfCNN 10	0.29/11.15	0.47/11.65	0.41/15.11	0.53/14.10	0.45/12.52	0.25/14	0.29/11.20	0.38/12.81
SurfCNN 1	0.31/11.57	0.5/12.10	0.42/17	0.56/16.52	0.53/13.20	0.26/18.19	0.26/18.19	0.41/14.54
PoseNet	0.32/8.12	0.47/14.4	0.29/12.0	0.48/8.42	0.47/8.42	0.59/8.64	0.47/13.8	0.44/11.63
Posenet-U	0.37/7.24	0.43/13.7	0.31/12.0	0.48/8.04	0.61/7.08	0.58/7.54	0.48/13.1	0.46/9.81
G-Posenet	0.20/7.11	0.38/12.3	0.21/13.8	0.28/8.83	0.37/6.94	0.35/8.15	0.37/12.5	0.31/9.94
BranchNet	0.18/5.17	0.30/7.05	0.27/5.10	0.33/7.40	0.38/10.30	0.20/14.20	0.34/8.99	0.28/8.37
Pose-L	0.24/5.77	0.34/11.9	0.21/13.7	0.30/8.08	0.33/7.00	0.37/8.83	0.40/13.7	0.31/9.85
Pose-H	0.15/6.17	0.27/10.8	0.19/11.6	0.21/8.48	0.25/7.01	0.27/10.2	0.29/12.5	0.23/9.5
VidLoc	0.16	0.21	0.14	0.24	0.36	0.31	0.26	0.25
RelocNet	0.21/ 10.9	0.31/10.3	0.40/10.9	0.33/10.3	0.33/11.4	0.15/13.4	0.32/11.8	0.28/11.28
Mobile-PoseNet	0.19/8.22	0.37/13.2	0.18/15.5	0.27/8.54	0.34 /8.46	0.31/8.05	0.45/13.6	0.30/10.79

The median position/orientation error for the U-SURF descriptors of features sizes

300, 100, 50, 10, 1 along with the state of the art previous work including PoseNet [1], G-Posenet [4], Posenet-U [5], Pose-L [2], BranchNet [6] Pose-H [3], VidLoc [7], RelocNet [8] and Mobile-PoseNet [9] for the 7 scenes dataset are shown in Table 2.3. As shown, with 300 U-SURF features, our system displays 0.28 m position error and 9.17° orientation error. Compared to the previous works, the accuracy of our model is comparable but we are advanced as to adopt only half the number of parameters and much smaller input dimension. Among all other models, Pose-H [3] demonstrates a better accuracy with 5 cm error margin whilst Pose-H employs an enormously complex network and needs to be trained seperately in every scene. VidLoc achieves 0.25 m positional error but with using up to 400 frames which is very hard to feed to the network with frame size of $224 \times 224 \times 3$ and require very expensive computational power. Further when the number of features is reduced to as low as 1 feature with training time around 15 minutes, an average positional and orientation error of 0.41 m and 14.54° is achieved. This is better than PoseNet and Posenet-U with an order-of-magintude reduction in the input and number of parameters. The error is acceptable in indoor environments while the input dimension is only 1×64. This opens the door for on-line training or training on embedded systems as the agent with the camera mounted on it has to train every time it goes to a different scene.

Among all individual scenes, our system outperforms all the previous work in the scenes of heads and fire. This is because the images in these scenes have the distinctive features and low field of view as shown in Fig. 2.2. Consequently, as few as 200 features with input dimension 200×64 is needed to outperform all the previous work that used input size of $224\times224\times3$ after cropping the input images. In the stairs scene, the system has very close performance to [3]. By analyzing the histogram of the stairs scene in Fig. 2.1, it is found that the maximum number of features in this scene is 500, which makes choosing 300 features sufficient for learning. However,

the chess, office and pumpkin scenes are wide view angle images and the number of features is large. Therefore, 300 features are insufficient to represent the input. Nevertheless, in these 3 scenes, SurfCNN still has a median error that is acceptable in indoor localization and competitive to the other work with substantial reduction in input dimension and lower training time.

We further demonstrate that our model is valid for different datasets other than the 7 scenes dataset, by comparing with PoseNet [1]. Firstly, we validate our system with the RGB-D SLAM Dataset and other public datasets. We also extend this work on our locally generated data using a robot mounted with multiple sensors including camera, LIDAR, odometer and SONAR to collect RGB images with ground truth poses. As shown in Table 2.4, the median translational/rotational error are consistent with the previous results of the 7 scenes dataset and our system outperforms PoseNet in all scenes.

Table 2.4: The median translational (m)/ rotational (degrees) error for the RGB-D dataset (fr1/xyz, fr2/xyz, fr1/rpy and fr2/rpy and our own measured data

Dataset	**Training**	**Testing**	**SurfCNN**	**PoseNet**
fr1/xyz	798	1017	0.13/0.10	0.18/0.14
fr1/rpy	720	970	0.03/0.39	0.10/0.50
fr2/xyz	3663	3736	0.05/0.03	0.12/0.12
fr2/rpy	3290	3462	0.02/0.10	0.10/0.14
Our data	1300	500	0.40/6.50	0.60/8.30
Average			**0.12/1.42**	0.22/1.84

We further extend the comparison with traditional classical systems that do not depend on learning. Here, SurfCNN is compared with two state of the art systems, the feature based method ORB-SLAM [31] and the direct method LSD-SLAM [26]. As shown in Table 2.5, for the scenes from ICL-NUM dataset and RGBD SLAM benchmark, SurfCNN on average outperforms both ORB-SLAM and LSD-SLAM. This is also in agreement with the comparison with the deep learning-based methods. Moreover, SurfCNN can still work with a low number of features as shown in Table

5 while other feature matching methods like ORB-SLAM will fail to match images with low number of features. Also, SurfCNN uses a sparse representation of the image which will be faster than minimizing the photometric error using all the image pixels done by LSD-SLAM.

Table 2.5: The median error of position (m) for ICL-NUIM dataset (office room 0, 1 and 2), the RGB-D dataset (RGBD-1: fr3/long office household) compared to SurfCNNfor ICL-NUIM dataset (office room 0, 1 and 2), the RGB-D dataset (RGBD-1: fr3/long office household) compared to SurfCNN

Dataset	**SURFCNN (m)**	**ORB-SLAM (m)**	**LSD-SLAM (m)**
ICL/office 0	0.45	0.43	0.52
ICL/office 1	0.50	0.76	0.78
ICL/office 2	0.52	0.79	0.68
RGBD 1	1.50	1.20	1.80
Average	**0.75**	0.80	0.95

2.5 Conclusion

We have implemented image based indoor localization (SurfCNN) that uses SURF descriptors to reduce the input dimension of CNN. Taking advantages of both SURF descriptors and CNN, our network has a competitive performance with the state of the art without the need for a pretrained network. Given a sufficient number of features, SurfCNN reaches the same accuracy as state of the art with only half the parameters and excluding the pretrained network. This advantage is essential in real time application where memory size is limited and edge/cloud computing applications. The proposed approach is also versatile to other CNN related applications in addition to indoor localization.

Chapter 3

SURF-LSTM: A Descriptor Enhanced Recurrent Neural Network For Indoor Localization

3.1 Introduction

In this chapter, we extend the idea presented in Chapter 2 of using speeded up robust features (SURF) [17] descriptors with neural networks for single image pose estimation. Traditional visual localization algorithms rely on extracting hand crafted features from images (e.g., scale-invariant feature transform (SIFT) [27] and SURF), along with keyframe matching and bundle adjustment optimization [65] to find the camera locations [31]. The performance of these methods relies on having a good initialization and correct matching between images features. Other visual localization approaches rely on 2D-3D matching where the image features are matched with the 3D model of the scene reconstructed from traditional methods such as structure from motion (sfm) [66]. These matchings are used to find the camera pose (position and

orientation) using n-points algorithm inside a Random sample consensus (RANSAC) loop to reduce the number of the outliers [48, 67]. Image retrieval-based algorithms use a database of images along with their absolute pose to approximate the query image pose by matching descriptors or bag of visual words representation with the database [68]. These methods have the limitation of the database size which increases with the scene area and also the matching process can be slow and prone to errors. Recently, machine learning algorithms are used for visual localization where neural networks are employed for absolute pose regression [1–3] and scene coordinate regression [69]. Instead of matching the handcrafted features with the 3D points from the reconstructed models, scene coordinate regression methods learn the 3D points corresponding to every pixel in the image using autoencoder neural networks [70]. Despite the acceptable accuracy reached by these methods, their training procedures are highly complex and require 3D data [71]. Absolute pose regression neural networks learn the pose of a single image by adding additional layers to pretrained models such as ResNet [36] or GoogleNet [55] and train the whole model in an end to end fashion. As the absolute pose of the image is learned, the model needs to be trained or fine-tuned again if the image belongs to another environment. This poses a problem especially with the complex training process of these methods.

Our proposed work aims to reduce the training complexity in image based localization for practical applications. Instead of using the whole image with pretrained networks which takes hours to train, we propose SURF-long short term memory (SURF-LSTM) that uses the strongest SURF features descriptors [17] of an image with only 2 layers of recurrent neural network (RNN) to learn the pose of the original image. SURF-LSTM can be trained in less than 10 minutes, suitable for the absolute pose regression application in which training is done on every different scene with comparable results to state of the art. Given an input image, SURF features are

extracted and we choose the strongest features to compute their descriptors of size 64 that describe the small area around every feature. Using the SURF features, we do not have to crop the image to a certain size to be suitable for the pretrained networks which can lose some important information to localization. Instead, we focus on the strongest features wherever they are in the image without cropping and reduce the input size from $224 \times 224 \times 3$ to $N \times 64$ where N is the number of features. Next, we use the power of LSTM network [56] to model the relation between the strongest features descriptors and the pose of the original image.

3.2 Localization Method

Our goal is to build a low complexity and fast neural network for image pose estimation that can be trained or fine tuned with low computational power and fast training time. We rely on LSTM to find the relation between the strongest SURF descriptors of the input image. This approach will save both time and memory compared to other CNN based approaches which depend on very large networks working on a full image.

3.2.1 SURF Descritpors

For an input image I, we use SURF algorithm [17] to extract the most significant keypoints along with their descriptors in the image. SURF uses a blob detector based on Hessian approximations to find keypoints as shown in (3.1). The Hessian matrix at a point $\mathbf{x} = (x, y)$ in the image is found by convolving the image with the second order derivative of a Gaussian filter g with a standard deviation σ as given by (3.2)

and (3.3).

$$H(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma), L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma), L_{yy}(\mathbf{x}, \sigma) \end{bmatrix} \tag{3.1}$$

where

$$L_{xx}(\mathbf{x}, \sigma) = I(\mathbf{x}) * \frac{\partial^2}{\partial x^2} g(\sigma) \tag{3.2}$$

$$L_{xy}(\mathbf{x}, \sigma) = I(\mathbf{x}) * \frac{\partial^2}{\partial xy} g(\sigma). \tag{3.3}$$

The detection process depends on non-maximal-suppression of the determinant of the Hessian matrices approximated using integral images obtained by cumulative addition of intensities on subsequent pixels in both horizontal and vertical axis instead of the original images and box filters as an approximation of the Gaussian filter second order derivatives. Hence the keypoint extraction process depends on the determinant of the approximated Hessian given by

$$Det(H_{approx}) = D_{xx}D_{xy} - (0.9D_{yy})^2 \tag{3.4}$$

where D_{xx}, D_{xy} and D_{yy} are the box filter approximations. The determinant of a Hessian matrix is an indication of the response of the keypoint (feature) and the higher the value, the stronger the feature is. Further, for the matching process between images, a descriptor for each feature is computed to describe the area around it. The Haar-wavelet transform is computed in both the horizontal and the vertical directions of a 4×4 grid around the feature. The descriptor vector of length 64 is computed using the sum of the HAAR responses in the horizontal and vertical direction. The computation can be done along the direction of dominant orientation of the grid around the feature which will produce the ordinary SURF descriptors that are invariant to rotations. Moreover, we can compute the descriptors with neglecting

the dominant orientation calculation to make them rotationally variant which are called upright SURF. We use only the upright SURF descriptors (for simplicity we call them SURF descriptors) in our work following [10] that states the advantages of using upright SURF over the ordinary SURF features for single image localization. As the U-SURF descriptors skip the dominant orientation calculation, the descriptors will not be oriented along the direction of the dominant orientation which will help finding the orientation of the image itself.

3.2.2 Problem Formulation

Given an input image I, multiple SURF features are extracted, each feature k_i is associated with its descriptor d_i of length 64. We choose the strongest N features using (3.4). We feed the strongest N features descriptors to bidirectional LSTM to learn the relation between these descriptors and use it to regress the pose P of the input image as

$$P = f_\theta([d_1, d_2, ..., d_N]) \tag{3.5}$$

where P consists of the position $[x, y, z]^T$ and the orientation in quaternion form $[q_w, q_x, q_y, q_z]^T$ of the image, f_θ is the bidirectional LSTM with learning parameters θ and d_i being the descriptor of the i^{th} feature with a total number of N features.

3.2.3 Architecture

We now go into the details of the architecture described in (3.5) where the strongest N features descriptors of the input image $[d_1, d_2, ..., d_N]$ sorted by the strongest to the weakest response are fed to the bidirectional LSTM. RNN is well known for its ability to learn the sequential relation between input instances and we use this capability to learn the relation between the N strongest features of the input image. We choose

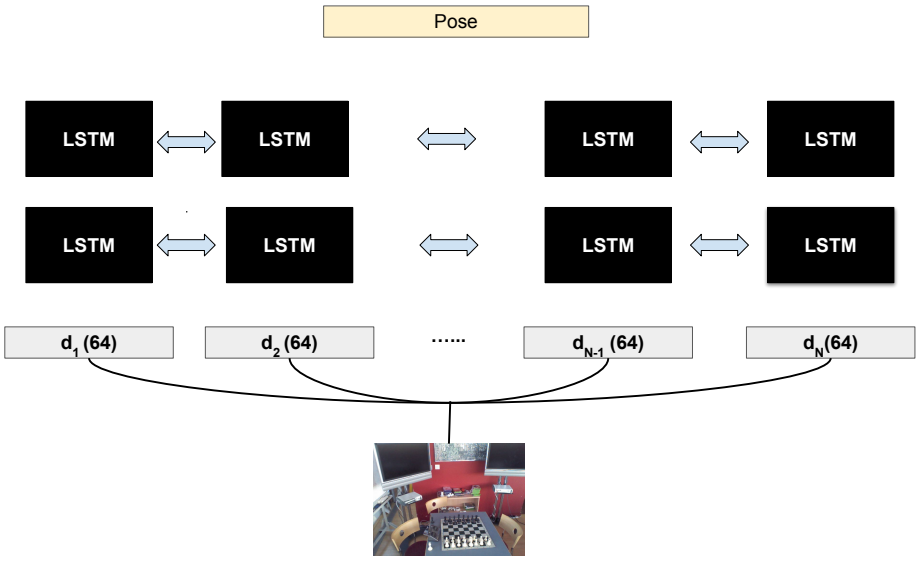


Figure 3.1: SURF-LSTM architecture.

LSTM instead of simple RNN for LSTM's ability to reduce the effects of the vanishing gradient problem [56]. LSTM contains four gates (input, output, memory and forget gate) and the gates are updated from time $t-1$ to time t as

$$\begin{aligned} i_t &= \tanh(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\ j_t &= \sigma(W_{xj}x_t + W_{hj}h_{t-1} + b_j) \\ c_t &= f_t \otimes c_{t-1} + i_t \otimes j_t \\ h_t &= \tanh(c_t) \otimes o_t \end{aligned} \tag{3.6}$$

where x_t is the current input, i, f, o, c are the input, forget, output and memory gate vectors respectively, j is an intermediate output that is used on the memory gate c and W, h, b are the weights, hidden state and bias. Further, as noted in the update equations (3.6) of LSTM, it uses the previous hidden state (or states depend on the

forget and memory gates) to predict the current output so that learning is done in one direction. Bi-directional LSTM [18] relies on the same concept of updating the gates of LSTM, however learning is done with the contribution of both past and future states which will help in modelling the relation between the descriptors in the direction from the strongest to the weakest (forward) and opposite direction (backward). As shown in Fig. 3.1, we use two bidirectional LSTM layers and each layer contains 100 recurrent units. The learned sequential features are then employed to find the pose of the image. We adopt the L_2 loss function to train the network written as

$$loss(D) = \left\| \hat{P} - P \right\|_2 \tag{3.7}$$

where D is the image descriptors set, $\hat{P}$ is the predicted pose, and P is the ground truth pose. Adam optimizer [64] is used to optimize the loss function with a learning rate of 0.0001. The training is done on NVidia TITAN-X GPU.

Table 3.1: The number of layers and learning parameters.

Network	Layers	Pretrained Network	Pretrained Parameters	Total Parameters
PoseNet	24	GoogLeNet	1.10×10^7	2.35×10^7
Pose-LSTM	28	GoogLeNet	1.10×10^7	2.15×10^7
SurfCNN	7	None	0	1.31×10^7
SURF-LSTM	2	None	0	3.73×10^5

3.3 Datasets

Similar to Chapter 1, We use the 7 scenes dataset and TUM RGBD dataset for training and testing our system. For TUM RGBD dataset, we choose to work with 4 different scenes ranging from simple scenes (fr1/xyz and fr2/xyz) to medium complexity scene (fr3/nonstructure texture near with loop) and finally with large scene (fr3/long office household).

3.4 Complexity analysis

Our main contribution is that our system is lower in complexity and faster than other neural networks-based systems. In this section, we do complexity analysis in terms of

- Training and testing time.
- Number of network parameters.
- Storage size of the image frame and the weights file.

We start by analyzing the training and testing time. Working with 50 SURF descriptors or less downsamples the input image by more than 47 times compared to working with the cropped images of size $224 \times 224 \times 3$ used by other works [1, 2]. Moreover, we use only two layers network which makes the training and testing process much faster with less memory requirements. In Fig. 3.2 (left), we show the training and testing time of SURF-LSTM compared to SurfCNN [10], PoseNet [11] and Pose-LSTM [2]. SURF LSTM achieves 9.8 minutes average training time over all number of features which reduces the training time by 14 minutes, 12.5 hours and 13.7 hours compared to SurfCNN, PoseNet and Pose-LSTM respectively. This huge reduction in training time helps the implementation of our system on any platform whether it is at the edge or server side. Moreover, because of using the down sampled descriptors, the testing time per frame is also lower for SURF-LSTM compared to state of the art as shown in Fig. 3.2 (right).

As our network consists of only two bidirectional LSTM layers and output layer, the number of parameters is reduced a lot compared to the state of the art. We compare the number of learning parameters for SURF-LSTM, SurfCNN, PoseNet, and Pose-LSTM in Table 3.1. SURF-LSTM does not depend on pre-trained networks

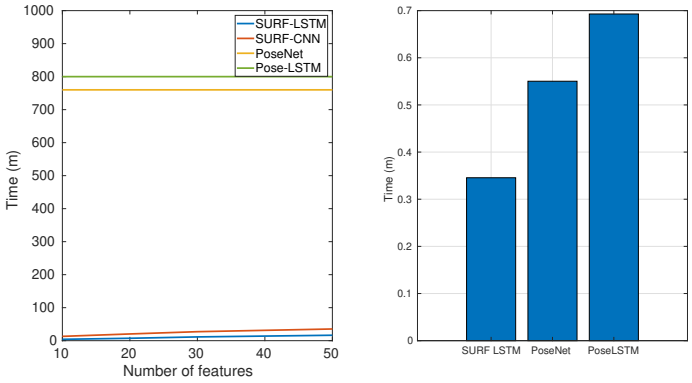


Figure 3.2: Training time (left) and testing time (right).

like SurfCNN but with 35 times less learning parameters and more than 57 times lower than other works. Although comparing the number of layers between SurfCNN and SURF-LSTM seems to be not fair as they are different types of neural networks, the number of layers affects the number of parameters as more layers will add more parameters to learn. So the number of parameters is the most important variable to consider when comparing between different types of neural networks as it will affect how fast the network will learn and whether the network will overfit on the training data or not.

Reducing the number of learning parameters is very beneficial in multiple ways, as it reduces the effects of overfitting, especially when we do not have enough training samples which is the case for the image based localization. It also reduces the space needed to store the weights file. The storage space is very critical in the robotics and IOT applications where the storage space is limited at the edge side. Moreover, it is important to have small size files if we want to train the network at the remote server or cloud side for easy transmission of files through the internet. We compare the storage size needed to store the image frame and the weights file in Fig. 3.3. As shown, only 0.0128 MB is needed to store the strongest 50 features descriptors of one image. This is 6.7 times lower than the cropped image used by other networks of size

$(224 \times 224 \times 3)$ and 30 times lower than the original image. Further, the weights file of SURF-LSTM needs only 1.5 MB compared to 50 MB in PoseNet plus the pretrained initialization weights of another 50 MB which means a total of 98.5 MB storage saved.

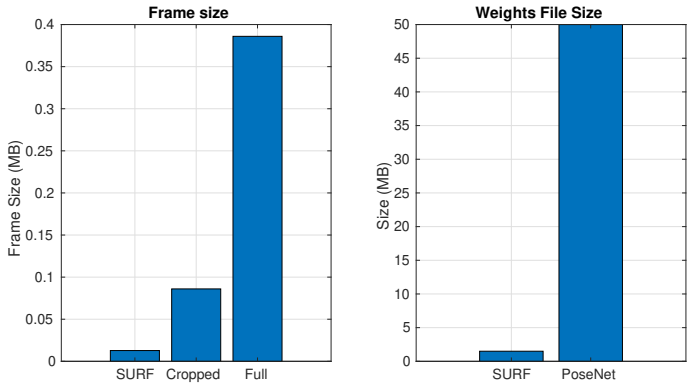


Figure 3.3: Storage size of image frames and weights file.

Table 3.2: The median error in position (m)/ orientation (degrees) for the Microsoft RGB-D 7 scenes dataset, with 7 input sizes from 10×64 to 300×64 of SURF-LSTM, compared with SurfCNN [10], PoseNet [1], G-Posenet [4], Posenet-U [5], Pose-L [2], [6] and G-PoseNet [11], BranchNet [6] and Mobile-PoseNet [9]

Algorithm	**Chess**	**Office**	**Pumpkin**	**Kitchen**	**Stairs**	**Heads**	**Fire**	**Average**
SURF-LSTM 300	0.22/7.89	0.32/7.14	0.41/9.97	0.38/7.62	0.36/10.12	0.17/12.15	0.22/11.87	0.29/9.60
SURF-LSTM 100	0.21/7.87	0.31/7.04	0.40/9.99	0.36/7.64	0.36/10.15	0.17/12.19	0.23/11.85	0.29/9.53
SURF-LSTM 50	0.21/7.98	0.32/7.05	0.40/9.94	0.36/7.60	0.35/10.10	0.16/12.1	0.22/11.87	0.29/9.39
SURF-LSTM 40	0.22/8.05	0.32/7.00	0.38/9.45	0.36/8.94	0.37/11.45	0.18/12.85	0.22/11.85	0.30/9.95
SURF-LSTM 30	0.21/8.00	0.37/7.50	0.35/7.93	0.43/9.84	0.38/12.49	0.20/13.45	.25/12.00	0.31/10.23
SURF-LSTM 20	0.208/7.74	0.40/8.53	0.37/10.75	0.48/10.74	0.42/13.56	0.24/14.34	0.26/12.45	0.33/11.15
SURF-LSTM 10	0.26/9.09	0.42/9.50	0.38/11.93	0.55/11.98	0.44/13.94	0.26/15.56	0.29/13.94	0.37/12.24
PoseNet	0.32/8.12	0.47/14.4	0.29/12.0	0.48/8.42	0.47/8.42	0.59/8.64	0.47/13.8	0.44/11.63
Posenet-U	0.37/7.24	0.43/13.7	0.31/12.0	0.48/8.04	0.61/7.08	0.58/7.54	0.48/13.1	0.46/9.81
G-Posenet	0.20/7.11	0.38/12.3	0.21/13.8	0.28/8.83	0.37/6.94	0.35/8.15	0.37/12.5	0.31/9.94
BranchNet	0.18/5.17	0.30/7.05	0.27/5.10	0.33/7.40	0.38/10.30	0.20/14.20	0.34/8.99	0.28/8.37
Pose-L	0.24/5.77	0.34/11.9	0.21/13.7	0.30/8.08	0.33/7.00	0.37/8.83	0.40/13.7	0.31/9.85
Mobile-PoseNet	0.19/8.22	0.37/13.2	0.18/15.5	0.27/8.54	0.34 /8.46	0.31/8.05	0.45/13.6	0.30/10.79
SurfCNN	0.19/8.10	0.35/7.05	0.36/10.80	0.37/10.25	0.28/10.14	0.17/12	0.24/12.80	0.30/10.22

3.5 Performance Analysis

We start the performance analysis by showing the number of training and testing images along with the median position and orientation error for the 7 scenes dataset in

Table 3.2 compared to the state of the art visual localization methods. SURF-LSTM achieves an average error of 0.29 m using only 50 features which is slightly better than SurfCNN [10] that uses 300 features or comparable to PoseNet [1], PoseNet-U [5], Pose-LSTM [2], Mobile-PoseNet [9] and BranchNet [6] that use cropped images with CNN pretrained networks. Moreover, an average orientation error of 9.93° for the same number of features is on bar with all the other works but with much smaller total input size 50×64 and with huge reduction in memory requirements as mentioned in the last section. We also notice that the best number of features varies with scenes. For example, using 20 features gives the lowest error for the chess scene while 50 features is the case for the heads scene without further enhancement in performance if the number of features increased. To explain this phenomenon, we plot the average response expressed in (3.4) of the strongest 100 features for some of the scenes in Fig. 3.4 with the best number of features stated on the legend of every curve. We note that there is a relation between the responses of features and the best number of features. So for example, in the chess scene, the responses of the strongest 20 features are very large which means that the features are very distinctive and therefore they are enough for localization. However in other scenes like the heads scene, the responses of the features are lower and therefore more distinctive features are needed to achieve the best performance. Further, one reason why adding more features does not improve performance is that the change in the response curve becomes small after certain number of features which indicates that the responses of the extra features are very close which will not have significant sequential relation to be learned by the LSTM.

We validate our system using another indoor dataset (TUM RGBD dataset) in Table 3.3. Median transnational error in meters of SURF-LSTM is compared with non-neural networks based systems, a features-based method (ORB-SLAM) and a direct method (LSD-SLAM) for 4 scenes of the TUM dataset. As noted from Table

3.3, our system outperforms both systems with an average error of 0.41 m without the need to do optimization on the whole image or initialization and matching between images.

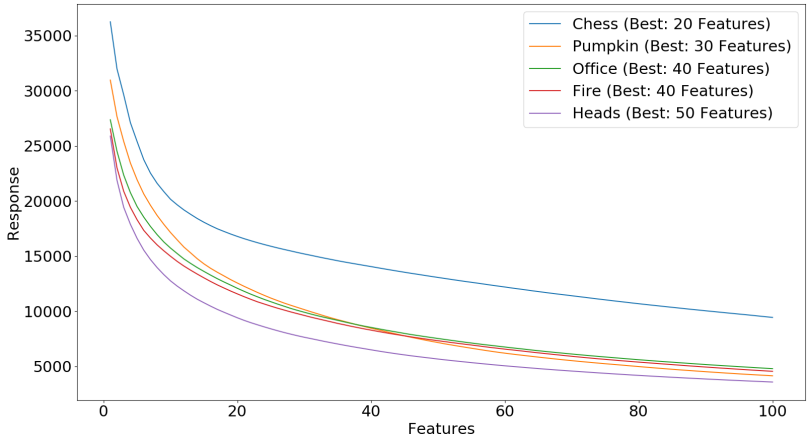


Figure 3.4: The response of the strongest 100 features of multiple scenes of the 7 scenes dataset.

Table 3.3: The median translational (m)/ rotational (degrees) error for the TUM RGBD dataset (fr1/xyz, fr2/xyz, fr3/long office household and fr3/nostructure texture near withloop.

Dataset	**ORB-SLAM**	**LSD-SLAM**	**Surf-LSTM**
fr1/xyz	0.08	0.10	0.06
fr2/xyz	0.04	0.06	0.03
fr3/long office household	1.21	1.85	1.12
fr3/nostructure texture near withloop	0.436	0.49	0.45
Average	0.45	0.625	0.41

3.6 Design analysis

In this section, we discuss the choice of SURF descriptors and Bidirectional LSTM in our system. Firstly, there are many features detectors and descriptors other than SURF including SIFT, ORB, FREAK and BRIEF. It is shown in [40] that the highest

accuracy in image matching for indoor localization is achieved using SURF descriptors. Further, we show in Fig. 3.5 the median positional error (m) for the heads scene of the 7 scenes dataset for various number of features descriptors. As shown, using SURF descriptors outperforms all other descriptors by a noticeable margin which validate the advantage of using SURF descriptors instead of the other descriptors. Secondly, we use RNN to find the relation between the strongest descriptors of an image. There are multiple types of RNN including simple RNN, Gated Recurrent Unit (GRU) [72], LSTM and the bidirectional version of them. Fig. 3.6 shows the comparison between all RNN types for the heads scene for various number of features. Among all of the RNN different types, bidirectional LSTM achieves the best positional accuracy.

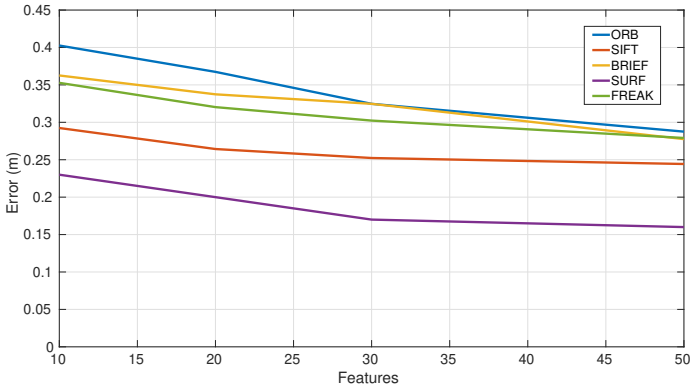


Figure 3.5: The positional error (m) of the heads scene for SIFT, SURF, ORB, FREAK and BRIEF descriptors for various number of features.

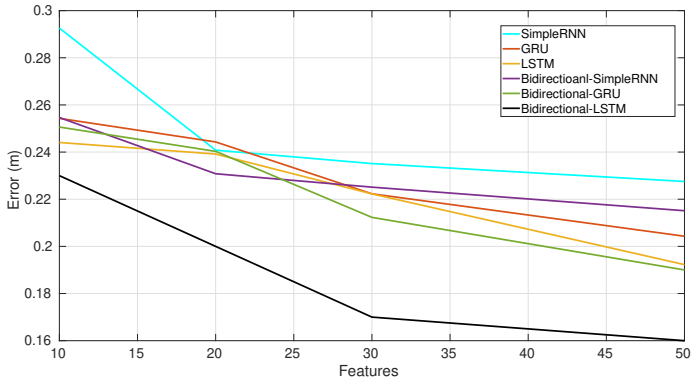


Figure 3.6: The positional error (m) of different RNN types for the heads scene with various number of features.

3.7 Conclusion

In this paper, we propose a fast neural network architecture for image based localization in indoor environments. The algorithm starts by finding the strongest SURF features descriptors of the image and use recurrent neural networks to find the relation between them to find the pose of the image. Our system reaches low positional and orientation error with orders of magnitude complexity reduction in terms of training time and storage and memory requirement.

Chapter 4

Pose-GNN : Camera Pose Estimation System Using Graph Neural Networks

4.1 Introduction

To enhance the performance of the pose estimation, machine learning techniques are used without the need for extracting hand crafted features or matching the images which is susceptible to errors and drift. Neural networks are used for the pose estimation tasks by either directly regressing the pose of the single image or by constructing the depth map of every pixel (scene coordinate regression). The latter methods can lead to similar or better results compared to the classical pose estimation methods but they require very complex training procedure and a ground truth depth map during training. This chapter belongs to the single image pose estimation system, however, instead of down-sampling the input images by using SURF descriptors used in Chapters 2 and 3, we use the image features extracted from a neural network pre-trained

on image classification task. Moreover, for the first time according to the authors' knowledge, we use the graph neural network (GNN) for image pose estimation. GNN is a class of neural networks that learns to do tasks by aggregating the features of graph structured data. A graph $G = (V, E)$ is defined as a set of nodes or vertices V and connections between these nodes or edges E. Each node is associated with features that are processed initially by GNN. The purpose of GNN is to update the node representation through multiple layers by aggregating the information of the neighbours of every node through message passing to have final hidden features for every node [73]. Nodes representation can be updated using recurrent neural networks (RecGNN) [74] or using multiple forms of convolution (ConvGNN) [75]. The final node representation can be used for many tasks including node level tasks where there is a label for every node such as semi supervised node classification [76] or graph level tasks where there is one label for the whole graph such as molecule property regression [77]. Single image pose estimation networks mostly use convolutional neural networks with pretrained initialization with the addition of more layers and using transfer learning to train the overall structure in an end to end fashion. We propose a new pipeline to this process, firstly, instead of training the pretrained CNN again, we show that the pretrained features are good enough for pose estimation task without further fine-tuning. Secondly, instead of using fully connected layers [1] or LSTM layers [14], we leverage the power GNN to process the pretrained features. We propose two novel approaches to model the image features with ConvGNN:

1. Image as a node (node-pose).

2. Image as a graph (graph-pose).

For the node-pose approach, the second to last layer features vector of pretrained ResNet50 [36] are extracted for every image which will be used as the representation of

every image as a node in a big graph which contains connections between neighbouring images based on the similarity between the pretrained features. The ConvGNN is trained to regress the pose of every node in the graph. The second approach deals with image as a graph where instead of extracting the flattend pretrained features, the intermediate feature maps of ResNet50 are extracted for each image and then converted into a graph structured data by connecting the nearest neighbours of the features and ConvGNN is used to to regress the pose of each graph.

We do extensive experiments using both indoor and outdoor datasets and we show the superiority of both systems compared to the state of the art and the advantages of using either approach for pose estimation.

4.2 Graph Neural network for image pose estimation

4.2.1 Convolutional graph neural network (ConvGNN)

ConvGNN relies on doing convolutions on the sparse graph structure which is different from the image convolution with defined grid structure. Convolution is usually done in two ways :

1. Spectral convolution (Spectral ConvGNN)

2. Spatial convolution (Spatial ConvGNN)

Spectral ConvGNN

For undirected graph with adjacency matrix A and a degree matrix D which is the sum of neighbours to every node $D_i = \sum_j A_{ij}$, we can use the normalized graph Laplacian matrix $L = I_n + D^{-0.5}AD^{-0.5}$ to approximate the graph convolution using Fourier

transform properties [73]. There are many implementations and approximations to Spectral ConvGNN such as GCN [76] and ChebNet [78]. In our experiments, we use GCN for it's simplicity and proven performance in multiple applications [79]. GCN follows the following node representation update :

$$X^{'} = \hat{D}^{-0.5}\hat{A}\hat{D}^{-0.5}X\theta \tag{4.1}$$

Where $X^{'}$ is the new nodes features representation, X is the current nodes features, $\hat{A} = A + I_n$ is the adjacency matrix with added self loops (node is connected to itself) and $\hat{D}$ is the node degree matrix where $\hat{D}_i = \sum_j \hat{A}_{ij}$.

Spatial ConvGNN

Instead of relying on the spectral theory to approximate the convolution operation, spatial methods try to use the same convention of convolution done on regular grid structures like images which is simple aggregation of neighbours such as addition, product, minimum and maximum. Many spatial ConvGNN implement different aggregation criteria including GraphSage [80], Graph Attention Networks (GAT) [81] and WL-GNN [82]. We use WL-GNN for implemtnting our systems as it has different set of weights for the central node than the neigbours which helps improving the performance according to [83]. WL-GNN has the following node update formula :

$$x_i^{'} = \theta_1 x_i + \sum_{j \in N(i)} \theta_2 x_j \tag{4.2}$$

where $x_i^{'}$ and x_i are the new and current representation of node i, θ_1 and θ_2 are the weights for the central node i and the neighbours j to node i lying in the set $N(i)$ of all neighbours to node i.

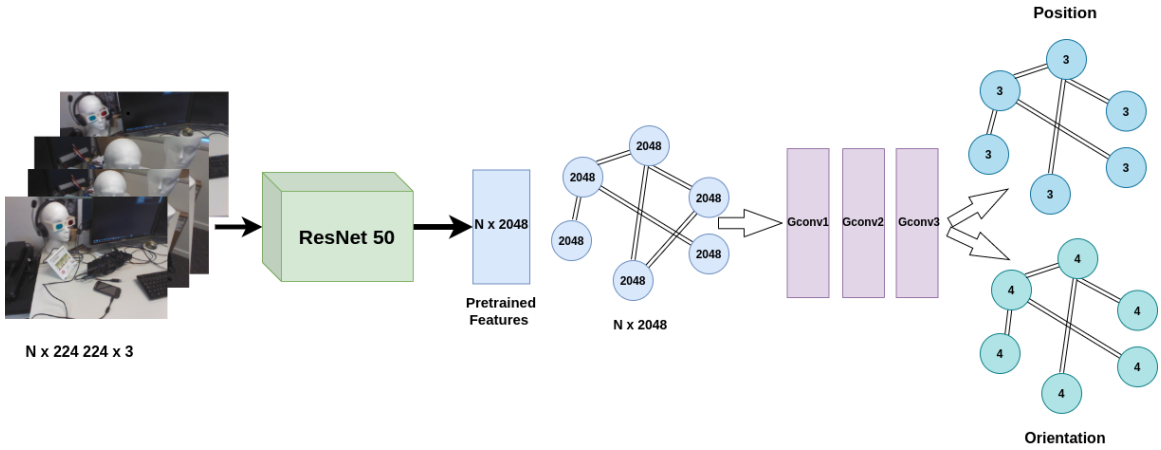


Figure 4.1: Image as a node architecture.

4.2.2 Image as a node (node-pose)

We show in Fig. 4.1 the details of the first architecture where images are modeled as nodes in a graph. To represent an image as a node, each image should be represented as a feature vector to use it as the initial features of the node. CNN architectures that are trained on a big image classification dataset for days are proven to act as a backbone to many applications with providing initial feature set to start with. Here, we use ResNet50 CNN architecture [36] that is widely used in many applications. Images are input to pretrained ResNet50 and the output of the second to last layer is obtained so each image will be represented by a feature vector of size 2048.

Training

For N training images with size $(N \times 224 \times 224 \times 3)$, pretrained features are extracted with total size $(N \times 2048)$. Following, the pretrained features representation is converted into a graph structure by building connections (edges) between images in the form of binary adjacency matrix. We use K-nearest neighbours (KNN) algorithm [19] to search for the nearest K neighbours of every image based on the L_2 distance between every image's pretrained features. For two images I_i, I_j with pretrained feature

representation x_i and x_j, the L_2 distance is calculated as

$$d(I_i, I_j) = d(x_i, x_j) = \sqrt{\sum_n (x_{in} - x_{jn})^2} \tag{4.3}$$

where n is the dimension of the pretrained features (2048). The images with smallest K distances are assigned as neighbours. Following, we use the feature matrix of the images $X \in R^{N \times 2048}$ and the binary adjacency matrix calculated using the KNN $A \in R^{N \times N}$ as input to the GNN. We use 3 Layers of graph convolutional layers (Gconv). The first layer accepts the node feature representation of size 2048 and learn new features of size 256 by aggregating the neighbouring features of each node according to (4.1). Following, new hierarchical features are learned in Gconv2 and Gconv3 of sizes 128 and 64 respectively that are used as input to the output fully connected layers of size 3 and 4 for position $p = [x, y, z]$ and orientation $q = [q_w, q_x, q_y, q_z]$ in quaternion form. We use the L_2 loss function to learn the pose of the images as :

$$loss(X) = ||\hat{p} - p||_2 + \alpha||\hat{q} - q||_2 \tag{4.4}$$

where α is used to balance the scaling between position and orientation. We choose the value of α to be 200 for outdoor environments and 10 for indoor environments.

Testing

After training is done and the weights of the model are saved, testing can be done for single or multiple images different than the training data. Specifically, for one or multiple testing images, pretrained features are extracted with size 2048, then, we find the K-nearest images from the training image set to construct a new graph containing the testing images and the K nearest training images by finding the L_2

distance between the pretrained features expressed in (4.3) . Following, the saved model will be used to aggregate information between the testing images and their neighbours and regresses the poses of the testing images.

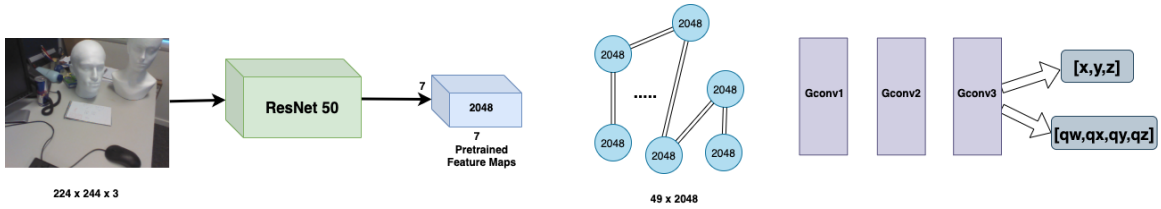


Figure 4.2: Image as a graph architecture.

4.2.3 Image as a graph (graph-pose)

Here, instead of dealing with images as nodes in a graph, we represent the images themselves as graphs. So if we have N images, we will have N small graphs instead of one big graph with N nodes. To represent the image as a graph, we need to decompose the image into sets of nodes and construct connections or edges between them. Dealing with the original image is not a feasible solution as it contains thousands of pixels that can be candidate nodes. In our system, we use ResNet50 pretrained CNN to extract feature nodes of the input image. All the images are input to ResNet50, then, instead of extracting the second to last flattened features, we extract the intermediate feature maps with general size $L \times W \times d$ denoting for the length, width and depth. These feature maps result from downsampling the input image through convolution, pooling and nonlinear activation function at the different layers with different number of convolutional filter. The details of the graph-pose system are shown in Fig. 4.2 where the input image is fed to ResNet50 and feature maps of the intermediate layers are extracted. ResNet50 consists of 50 layers; 1 flattened layer and 49 different feature maps that can be used as input to GNN. We use the feature maps from the layer before the flattened features with size $(7 \times 7 \times 2048)$. These

feature maps can be reshaped to (49×2048) which can be represented as a graph that contains 49 nodes with features vector of size 2048 for each of them. Next, edges between different nodes are constructed using the K-nearest neighbours with L_2 distance criteria between different nodes. For training this system, we used the same L_2 loss in (3.7).

4.3 Datasets

For testing the proposed systems in different conditions, in addition to the 7 scenes dataset presented in the previous chapters, we use the Cambridge dataset [11] which contains multiple scenes from outdoor environments with multiple images for training and testing along with the ground truth poses.

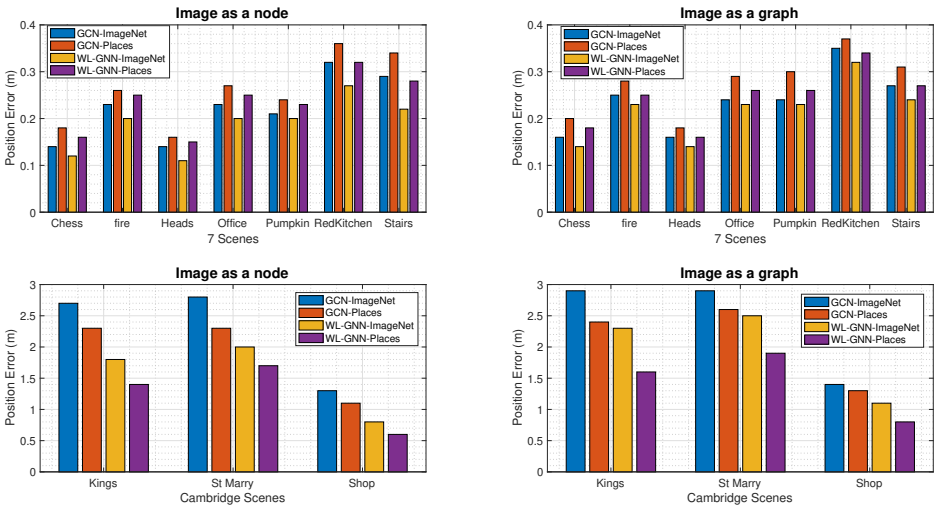


Figure 4.3: The effect of pretrained features and type of GNN on the position error for the 7 scenes and Cambridge datasets.

4.4 Design Analysis

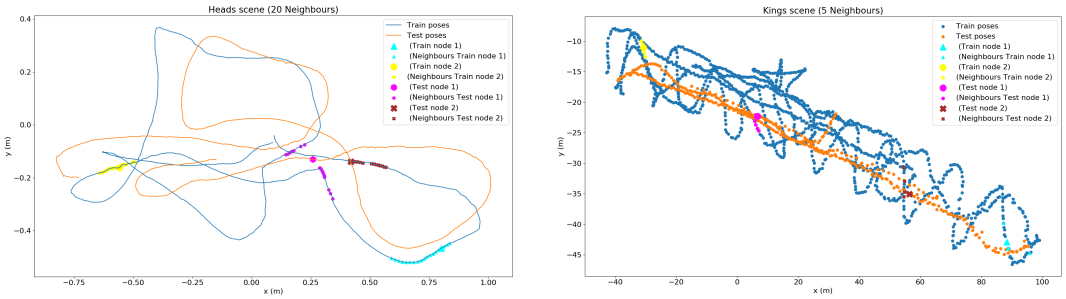
In this section, we discuss the effect of multiple design considerations on the localization accuracy including the choices of the pretrained dataset, the type of ConvGNN and the number of neighbours.

4.4.1 Pretrained features and types of ConvGNN

For both node-pose and graph-pose architectures, we use ResNet50 CNN to extract the features of the input images which is pretrained on large image dataset. We use ImageNet dataset [84] and Places dataset [85] as two pretrained datasets and we study the effect using both of them in Fig. 4.3 on the positional error of both the 7 scenes and Cambridge dataset. Moreover, the position error of two types of ConvGNN; spectral ConvGNN (GCN) and spatial ConvGNN (WL-GNN) is shown in Fig. 4.3. Multiple observations can be concluded from Fig. 4.3:

- Using ImageNet dataset always produces better results for indoor scenes than Places dataset.
- Using Places dataset gives lower error for outdoor scenes.
- WL-GNN outperforms GCN across all scenes and for both systems.
- Node-pose architecture performs better than graph-pose system.

Here, we discuss the above mentioned observations, firstly, ImageNet dataset is an object classification dataset that contains mostly one type of objects with simple scenes while Places dataset is a scene classification dataset with more complex scenes, many objects and mostly outdoor images. This can be one reason why ImageNet dataset pretrained features perform well for indoor scenes where the images contain few objects and scenes are not complex. On the other hand, Places dataset pretrained



(a) Heads Scene (b) Kings Scene

Figure 4.4: The connected positions for multiple training and testing nodes of Heads scene (left) and Kings scene (right).

features work well for outdoor scenes where the scenes are more complex and the images contain many objects. Secondly, we observe the superiority of WL-GNN over GCN for all scenes. One big difference between WL-GNN and GCN is that WL-GNN gives different set of weights to the central node than the neighbours in contrast to GCN that applies the same set of weights to all nodes. As mentioned at the benchmark paper [83], learning separate weights for central nodes is beneficial to the learning process and provide better results for multiple tasks and datasets which matches the current observation. Finally, node-pose system provides better results since multiple images are used as neighbours for the final decisions while only the current image information is used by pose-graph architecture without connecting to neighbouring images.

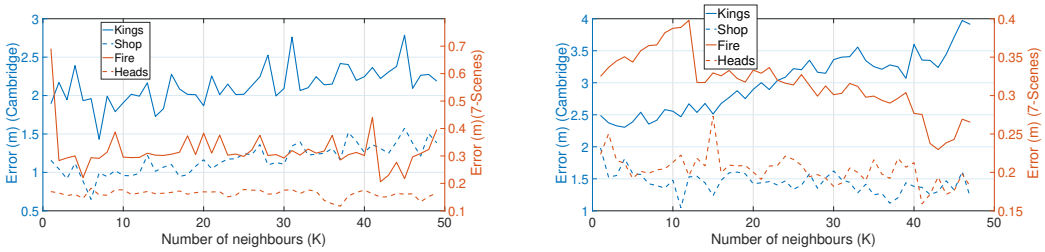
4.4.2 Effect of the number of neighbours

Both of the proposed GNN systems rely on aggregating the information between the central node and its neighbours where the number of neighbours is a hyperparameter we need to tune. We discuss in Fig. 4.5 the relation between the number of neighbours and median position error for the 7 scenes dataset (Heads and Fire scenes) on the right y axis and the Cambridge dataset (Kings and Shop scenes) on the left y axis for

both node-pose architecture (left) and graph-pose architecture (right). For clarity, the notion of neighbours differs for both systems. The neighbours for node-pose system are images with similar pretrained features, however, for pose-graph, the neighbours are similar elements from the pretrained features map. Firstly, for the node-pose system, we can see in Fig. 4.5 that for indoor Heads and Fire scenes, more than 35 neighbours are needed to reach the lowest positional error while less than 10 neighbours are used to achieve the best accuracy for the outdoor Kings and Shop scenes. One reason for such phenomenon is that the distance between consecutive images for the outdoor scenes is much higher than the indoor scenes. This means more neighbours for outdoor scenes will be far from each other which will not help localization. However, for indoor scenes, the camera moves very slowly and adding more neighbours up to a certain number will be beneficial and help localization as they will be close to the central node. Moreover, Fig. 4.4 shows the same observation that more neighbours for the indoor scenes are more consistent and closer than the outdoor scenes. In Fig. 4.4, 2 nodes from the training set and 2 nodes from the test set for each scene along with their neighbours from the training set are visualized. In the case of Heads scene, we can see that the positions of training nodes (in cyan and yellow) are very close which validate our claim that similar pretrained features of images leads to close positions. However, for the test nodes, as we find their nearest neighbours from the training set, we can see that the neighbours are not as ideal as the training nodes but they lie on the same area of the testing node. For the Kings outdoor scene, we can see that although the connected positions are not very far from each other, they are not also very close as the case for the indoor scene. This happens because the overlap between images is lower than the indoor scene and the distance between consecutive images is higher.

Secondly, despite the different neighbours representation for graph-pose system,

the same behaviour is present as shown in Fig. 4.5 where using more than 40 neighbours gives the best performance for indoor scenes and the lowest error for outdoor scenes is achieved using less than 10 neighbours. This behaviour is still preserved since for indoor images, the field of view of images is small which can enable us to relate more pixels or features together than the outdoor images whose field of view is very wide and lower features from the images can be related as neighbours.



(a) Image as a node (b) Image as a graph

Figure 4.5: The relation between the number of neighbours and the median position error (m).

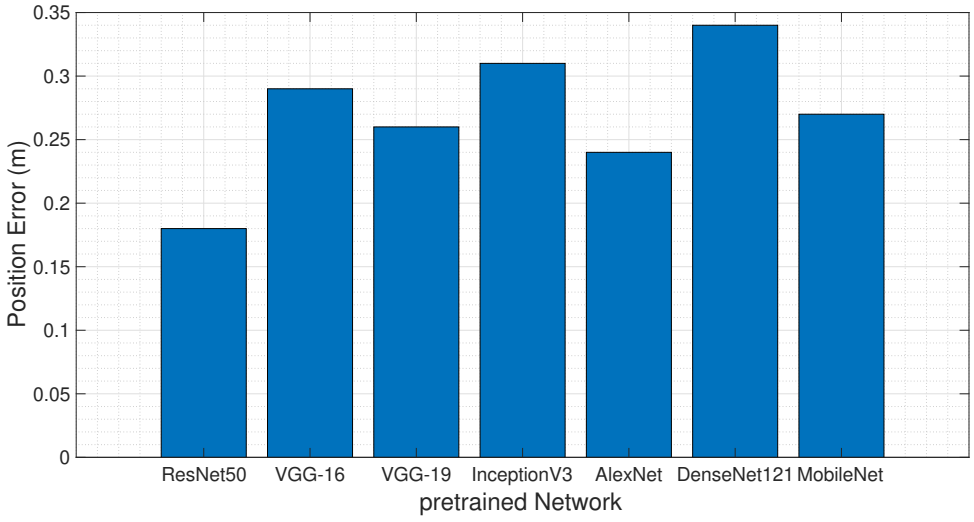


Figure 4.6: Average positional error of pretrained networks for the 7 scenes dataset

4.4.3 Effect of the pretrained network

In this section, we validate the choice of ResNet-50 as a backbone for our system. We show in Fig. 4.6 the average error for the 7 scenes dataset for different pretrained networks including VGG-16, VGG-19, InceptionV3, AlexNet, DenseNet 121 and MobileNet [86]. As shown, ResNet 50 produces the lowest average error with 0.18 m compared to other methods which have more than 0.24 m positional error.

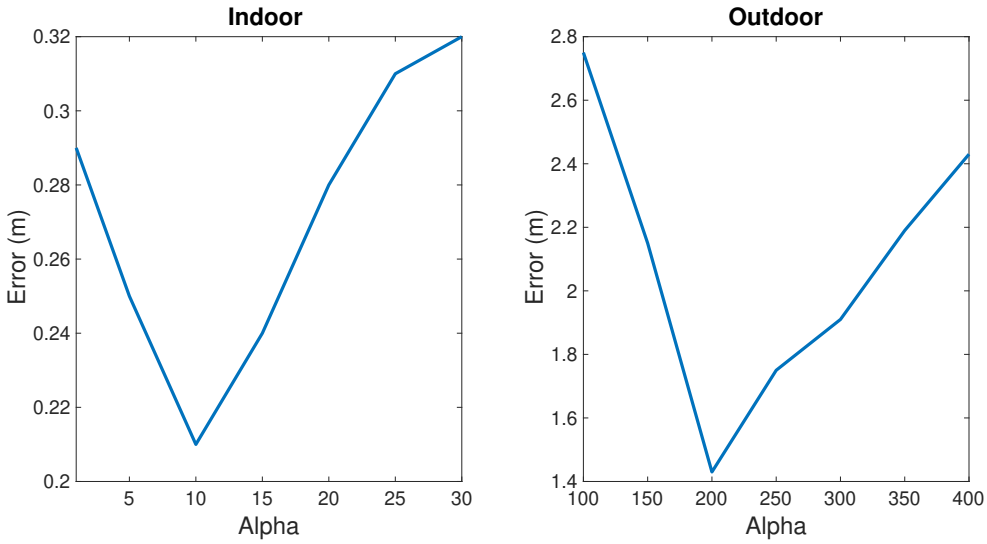


Figure 4.7: Average positional error of different values of Alpha for the 7 scenes indoor dataset and the Cambridge outdoor dataset

4.4.4 Effect of α values for indoor and outdoor scenes

Here, we study the effect of different values of α parameter presented in the loss function equation (4.4) on the performance. As shown in Fig. 4.7, the best value for α that produces the minimum average positional error in indoor scenes is about 10 while the value of 200 is the best for outdoor scenes.

4.5 Performance analysis

We validate the performance of the two proposed systems with the state of the art single image localization systems for all the scenes of the 7 scenes dataset and the Cambridge dataset interms of median position error (m) and median orientation error (deg) using both ImageNet and Places pretrained datasets in Table 4.1. As seen, node-pose system outperforms all the other methods for the 7 scenes dataset for both position and orientation reaching average error of $0.18m/7.5^{\circ}$ using ImageNet pretrained features. Moreover, node-pose outperforms all the other methods using Places pretrained features for the orientation calculation of the outdoor scenes with an average error of 3.76° and reach comparable results to PN L-W And LSTM. PN for the position error with only few centimeters difference and outperforming all the other methods. We can notice the same observation mentioned in Subsection 4.4.1 that using Places dataset is better for outdoor scenes while ImageNet pretrained features perform better for the indoor environments. For graph-pose system, we observe that although node-pose system outperforms it, graph-pose system still outperforms all the other methods for the 7 scenes dataset. However, for the Cambridge outdoor dataset, graph-pose architecture does not perform as well as for the indoor scenes but produce comparable results to the other localization methods.

Table 4.1: The median error in position (m)/ orientation (degrees) for the 7 scenes and Cambridge dataset for the proposed models, compared with SurfCNN [12], SURF-LSTM [13], PoseNet [1], G-Posenet [4], Posenet-U [5], Pose-L [2], [6] and G-PoseNet [11], BranchNet [6] ,Mobile-PoseNet [9],VidLoc [7] and Pose-Hourglass [3]

Algorithm	**Chess**	**Fire**	**Heads**	**Office**	**Pumpkin**	**Kitchen**	**Stairs**	**Average**	**Kings**	**St Marry**	**Shop**	**Average**
Node-Pose (ImageNet)	0.12/6	0.20/8.4	0.11/11.1	0.20/6.8	0.20/5.5	0.27/7.5	0.22/7.2	0.18/7.5	1.8/3.5	2/5.5	0.8/47	1.46/4.16
Node-Pose (Places)	0.16/8	0.25/9.4	0.15/11.8	0.25/8.1	0.23/6.5	0.32/9.5	0.28/9.2	0.23/8.9	1.4/2.8	1.7/5	0.6/3.57	1.2/3.76
Graph-Pose (ImageNet)	0.14/6.4	0.23/8.6	0.15/11.8	0.23/6.8	0.23/5.5	0.32/7.7	0.24/7.4	0.22/7.84	2.3/4.6	2.5/6.9	1.1/5.1	1.9/5.53
Graph-Pose (Places)	0.18/8.1	0.25/8.8	0.15/11.9	0.26/8.9	0.26/5.5	0.34/10	0.27/9.47	0.25/9	1.6/3.1	1.9/5.5	0.8/4	1.43/4.2
SURF-CNN	0.19/8.10	0.24/12.8	0.17/12	0.35/9	0.36/10.8	0.39/10.2	0.36/10.8	0.3/10.22	7.66/12.3	4.5/10.5	3.2/9.2	6.82/10.6
SURF-LSTM	0.22/7.04	0.24/10.2	0.16/13.6	0.35/8.5	0.35/7.8	0.38/9.5	0.35/11.9	0.3/9.4	5.53/10.6	3.2/9.2	1.46/8.1	5.37/9.3
PoseNet	0.32/8.12	0.47/14.4	0.29/12.0	0.48/8.42	0.47/8.42	0.59/8.64	0.47/13.8	0.44/11.63	1.92/5.40	1.46/8.1	1.11/7.6	2.01/7.03
Dense VLAD	0.21/12.5	0.33/13.8	0.15/14.9	0.28/11.2	0.31/11.3	0.30/12.3	0.25/15.8	0.26/13.11	2.80/5.75	1.11/7.6	1.25/7.5	2.07/6.95
Bay. PN	0.37/7.24	0.43/13.7	0.31/12.0	0.48/8.04	0.61/7.08	0.58/7.54	0.48/13.1	0.46/9.81	1.74/4.06	1.25/7.5	1.79/6.5	1.70/6.2
PN L-W	0.14/4.50	0.27/11.8	0.18/12.1	0.20/5.77	0.25/4.84	0.24/5.52	0.37/10.6	0.24/7.87	0.99/1.06	1.05/3.9	1.18/7.4	1.17/4.12
LSTM. PN	0.24/5.77	0.34/11.9	0.21/13.7	0.30/8.08	0.33/7.00	0.37/8.83	0.40/13.7	0.31/9.85	0.88/1.04	1.18/7.4	1.14/5.7	1.19/4.71
G PoseNet	0.20/7.11	0.38/12.3	0.21/13.8	0.28/8.83	0.37/6.94	0.35/8.15	0.37/12.5	0.31/9.94	0.99/3.65	1.14/5.7	3.2/9.2	1.68/6.18
Pose-Hourglass	0.15/6.17	0.27/10.8	0.19/11.6	0.21/8.48	0.25/7.01	0.27/10.2	0.29/12.5	0.23/9.5	NA	NA	NA	NA
Mobile-PoseNet	0.19/8.22	0.37/13.2	0.18/15.5	0.27/8.54	0.34 /8.46	0.31/8.05	0.45/13.6	0.30/10.79	1.14/1.53	2.18/6.1	1.73/6.19	1.6/4.60
Vidloc	0.16	0.21	0.14	0.24	0.36	0.31	0.26	0.25	NA	NA	NA	NA

4.6 Conclusion

We propose a novel image based localization system using graph neural networks (GNN). We use pretrained ResNet50 convolutional neural network (CNN) architecture to extract the important features for each image. Following, we use the extracted features as input to GNN to find the pose of each image by either using the image features as a node in a graph and formulate the pose estimation problem as node pose regression or modelling the image features themselves as a graph and the problem becomes graph pose regression. We do an extensive comparison between the proposed two approaches and the state of the art single image localization methods and we show that using GNN leads to enhancing the performance for both indoor and outdoor environments.

Chapter 5

Linear-PoseNet: A Real-Time Camera Pose Estimation System Using Linear Regression and Principal Component Analysis

5.1 Introduction

In this chapter, we explore the following question regarding single image pose estimation system : Do we need to build such big networks to reach good localization performance?. To answer this question, we propose Linear-PoseNet, a camera pose localization system that extend the idea used in Chapter 4 by using the features from the pretrained ResNet50 [36] CNN architecture without fine tuning them along with only one ridge regression layer [87]. Using this approach, our system can reach comparable performance to other single image pose estimation methods for indoor environments while reducing the input image size from 3D shape $224 \times 224 \times 3$ to

1D feature vector 2048 and training time in less than a second on CPU. This huge reduction in training time from hours needed by the other systems to less than a second and without the need for GPU can help implementing our system on any platform or hardware and training can be done on the fly. However, for complex outdoor environments, they require more complex network. We also show that the pretrained ResNet50 features are good enough to use in top of 3 fully connected layers without finetuning (Dense-PoseNet) to achieve good performance with training in few minutes. Moreover, for more dimensionality reduction of the input features, we emphasize that using principal component analysis (PCA) [88] to extract the principal components of the pretrained features does not degrade the performance significantly with more than 10 times reduction in the input size which will also save the training time and storage space.

5.2 Pretrained features extraction

Transfer learning [89] helps many applications to be built on top of the powerful CNN architectures. We can use the features learned by CNN layers trained on millions of images for image classification task as a backbone for a new application with the addition of more layers and training the whole system again in an end to end manner [90]. To train of the new network for a new task, there are two strategies :

1. Use the extracted features as they are (freeze the backbone network) and train only the newly added layers

2. Initialize all or part of the pretrained network layers with the pretrained weights and train the whole network.

Deciding which strategy to work with depends on the similarity between the pretrained task and the current application and whether the features are good enough

for the new task or not. Most of the previous work follow the second strategy where the intermediate layers of pretrained GoogleNet [35] is used with the addition of more layers. This strategy makes the training time and complexity very high. Here, we use ResNet50 CNN architecture [36] as a backbone network for its proven good performance on image classification [91] and other applications including image pose estimation as discussed in Chapter 4. For the pretrained dataset, we use the features of two of the most well known benchmarks in computer vision: ImageNet dataset [84] and Places dataset [85]. Moreover, we follow the first strategy, features from ResNet50 pretrained on ImageNet and Places datasets are used without any further training as the input to our models. In detail, we use the flattened features before the output layer of ResNet-50 with size 2048 so each image will be stored and further processed with these features.

5.3 System models

In this section, we discuss the details of the proposed systems for single image camera pose estimation shown in Fig. 5.1. Given an input image, I with pretrained features X, the objective is to learn the pose $P = [p, q]$ where the position $p = [x, y, z]$ and the orientation in a quaternion form $q = [q_w, q_x, q_y, q_z]$ of this image in an arbitrary reference frame as

$$P = f_\theta(X) \tag{5.1}$$

where f_θ is the neural network to be trained.

5.3.1 Linear-PoseNet

Given the pretrained features X extracted from the input image I, each image is represented as a 1D vector of size 2048. We propose using only one layer of linear

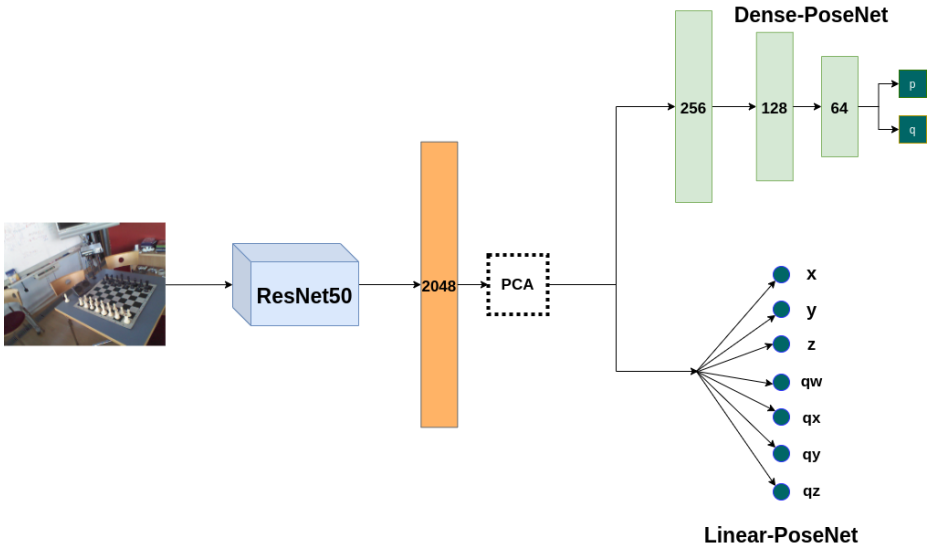


Figure 5.1: Linear and Dense-PoseNet architectures.

regression with the pretrained features to regress the pose of the image P. Specifically, the objective function of the Linear-PoseNet is to minimize the linear least squares with L_2 regularization known as ridge regression expressed as

$$||P - Xw||_2^2 + \lambda||w||_2^2 \tag{5.2}$$

where λ is a positive hyper parameter that relates to regularization strength and w is the learning parameters or the weights . The pose $P = [p, q] \in R^7$ and the features $X \in R^{2048}$ make the weights $w \in R^{2048\times7}$. We use ridge regression instead of the simple least squares as the regularization is considered as a norm penalty term which will lead to simpler solutions and prevent overfitting by constraining the values of w to lie on the L_2 ball [90]. The tuning of the hyperparameter λ is detailed in Section 5.5.

5.3.2 Dense-PoseNet

For complex environments such as outdoor scenes, we propose Dense-PoseNet that is composed of 3 fully connected layers of 256, 128 and 64 neurons. The architecture is designed by following experiments similar to Chapter 2 for the positional error and the number of layers. As shown in Fig. 5.1, the pretrained features X of the input image I are input to the 3 fully connected layers to learn the pose of the image. We use the L_2 loss function to learn the pose $P = [p, q]$ as

$$loss(X) = ||p - \hat{p}||_2 + \alpha||q - \hat{q}||_2 \tag{5.3}$$

where p, q are the ground truth position and orientation, $\hat{p}$, $\hat{q}$ are the predicted position and the normalized predicted orientation and α is a hyper parameter that scales the errors for orientation and position as they are in different scales. Based on our experiments similar to Chapter 4, We choose α to be 100 for big outdoor scenes and 10 for small indoor scenes.

5.3.3 Using PCA for dimensionality reduction

In this section, we explore the following question: can we further reduce the pretrained features dimensionality without affecting the performance dramatically? As discussed in Section. 5.2, we use the pretrained features of ResNet50 with size 2048 but do we need all these features for the pose estimation task? The dimensions of pretrained features is designed for the classification of the ImageNet and Places datasets which is different from the pose regression task that may need different number of features. The dimensionality reduction of features will save the training time and also reduce the storage and memory requirements needed to store the features. PCA is well known for features extraction and dimensionality reduction [92]. Here, we use

PCA to extract fewer features and study the effect of dimensionality reduction on the localization accuracy. The new downsampled feature set X' with K dimensions (principal components) is obtained by the linear projection of the pretrained features X as

$$X' = WX \tag{5.4}$$

where $W \in R^{K \times 2048}$ contains the eigenvectors corresponding to the largest K eigenvalues of the covariance matrix of the pretrained features X. The effect of number of principal components K on the performance is studied in Section 5.5.

5.4 Time and storage space analysis

We do experiments and validate the proposed systems using 7 scenes indoor dataset and Cambridge outdoor dataset used in Chapter 4. In this section, we show the saving in time and space achieved using the proposed systems compared to the other single image localization methods including SURF-CNN [12], SURF-LSTM [13], PoseNet [1], Pose-LSTM [14] and Pose-Hourglass [3] in Table 5.1. Specifically, for time analysis, we can see that Linear-PoseNet can be trained in only 0.46 seconds on CPU and this time can be further reduced by almost one half if we downsample the input features using PCA. This huge reduction in the training complexity compared to other methods which need significant training time on GPU suits the application of single image localization where we can train the system when the agent moves to new environments. Moreover, the simple training scheme encourages the implementation of the system on low-power and low specification hardware with no need for GPU, such as an embedded device. This allows edge computing not only implement a testing model, but also train/retrain the model as needed in an edge device. For

Table 5.1: Time and storage space comparison

Algorithm	**Training hardware**	**Training Time (minutes)**	**Testing time /frame (minutes)**	**Number of learning Parameters**	**Frame Size (MB)**	**Weights File size (MB)**
Linear-PoseNet	CPU	0.0076	7×10^{-6}	0.14336×10^5	0.0082	0.1615
Linear-PoseNet (PCA)	CPU	0.0046	7.5×10^{-7}	1960	0.0011	0.022533
Dense-PoseNet	GPU	3	0.25	5.73824×10^5	0.0082	2.2
Dense-PoseNet (PCA)	GPU	1.78	0.15	1.13×10^5	0.0011	0.472392
SURF-LSTM	GPU	12	0.35	3.73×10^5	0.03	1.5
SURF-CNN	GPU	20	0.45	1.31×10^7	0.08	1.5
PoseNet	GPU	720	0.65	2.15×10^7	0.095	50
Pose-LSTM	GPU	780	0.69	2.35×10^7	0.095	50
Pose-Hourglass	GPU	900	0.72	3.5×10^7	0.095	70

Dense-PoseNet, despite that it needs to be trained on GPU, we can see that it can be trained in only 3 minutes with significant reduction in time compared to the other methods. The training time can be reduced further by around 50 % using PCA as a preprocessing step. Not only the training time is saved by our systems, but also the testing time which includes loading the saved model, loading the image and predicting the pose. We can see in Table 5.1 that the testing time per image is substantially reduced by Linear-PoseNet with testing time in microseconds instead of nearly half a second for other methods. Dense-PoseNet reduces the testing time by 1.5 times or more per frame. For both systems, the testing time per frame can be further reduced with PCA downsampling. Here, we discuss the storage space comparison, the number of learning parameters or weights are shown in Table 5.1 where the Linear-PoseNet reduces the number of weights by more than 26 times compared to the other works and by more than 190 folds when PCA is used. Although Dense-PoseNet has comparable number of weights to SURF-CNN and SURF-LSTM, it can work in outdoor environments where SURF-CNN and SURF-LSTM are only limited to indoor scenes. Moreover, Dense-PoseNet has lower number of weights than PoseNet, Pose-LSTM and Pose-Hourglass and using PCA can make the learning parameters lower than all other methods by 2 times or more. For some cases, training or fine tuning the networks can not be done on the client side where the computation budget is limited. In this case, we need lower sizes of weights file and image frames to be sent to the server side where further processing is done. Here, we show in Table 5.1 that using our systems, the frame size can be reduced to only 0.0082 MB as we use features of the original images and can be sent to the cloud very easily. Compared to other methods, Linear-PoseNet and Dense-PoseNet need up to 11 times lower space to store the images. For the single image pose estimation application, the networks need to be fine-tuned again if new environment is encountered or new images are available.

Fine tuning requires using the available weights as initial weights for the new training procedure. So having small size of weights is beneficial. Only 0.1615 MB is needed to store the Linear-PoseNet weights with about 49.8 MB reduction compared to CNN-image based localization systems. The weights file size can be further reduced to 0.023 MB using PCA. Further, Dense-PoseNet needs 2.2 MB using full features which is comparable to SURF-CNN and SURF-LSTM and 0.47 MB using PCA features which is lower than all other methods.

5.5 Design Analysis

In this section, we discuss the design considerations and hyper parameter tuning procedure for our systems including :

1. Pretrained features
2. Regularization multiplier for Linear-PoseNet
3. The number of principal components for PCA

5.5.1 Pretrained features

In our system, we use the ResNet50 features pretrained on ImageNet and Places datasets. Here we discuss the effect of using either datasets on the performance of indoor and outdoor localization. Fig. 5.2 shows bar plot comparison of the position error in meters of the Linear-PoseNet (left) and Dense-PoseNet (right) of the 7 scenes and Cambridge datasets using ImageNet pretrained features (in orange) and Places pretrained features (blue). We can see that for indoor scenes of the 7 scenes dataset, using ImageNet features always produces lower error than Places features. However, the opposite is true for the outdoor scenes of the Cambridge dataset where Places

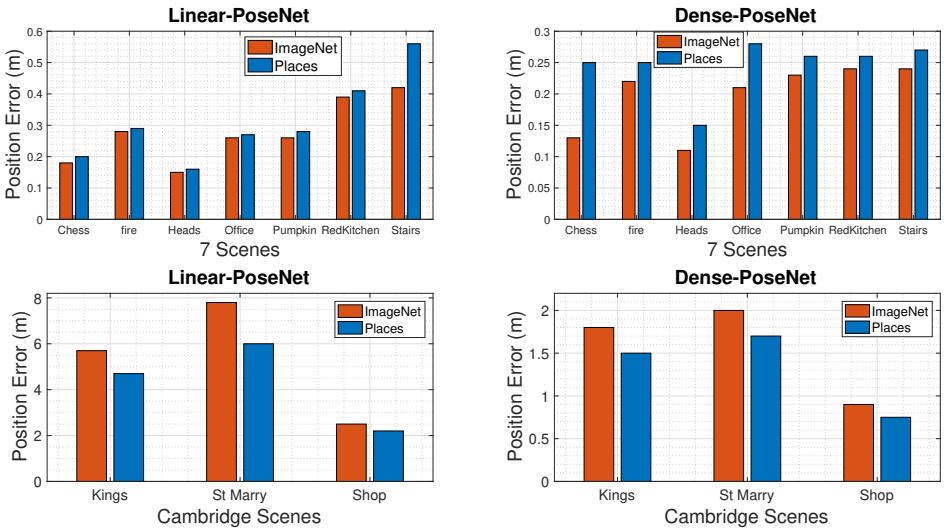


Figure 5.2: Effect of using ImageNet and Places datasets features on the performance of Linear-PoseNet and Dense-PoseNet.

dataset is better by 1 meter on average. One reason for such observation is that ImageNet dataset is an object classification dataset where each image mostly contains one type of objects and the image details are not complex which is similar to the indoor scenes in the 7 scenes dataset where each image contains few number of objects. However, Places dataset is a scene classification dataset with very complex images of many objects and many details and mostly outdoor scenes which explains the good performance for outdoor environments. So the conclusion is ImageNet pretrained features are good for simple indoor environments and Places pretrained features are suitable for more complex outdoor scenes.

5.5.2 Regularization multiplier for Linear-PoseNet

For Linear-PoseNet, we use the L_2 objective function described in (5.2) where there is a regularization hyperparameter λ which models the weight given to the L_2 regularization term. In Fig. 5.3, we show the effect of choosing different values of λ on the pose estimation error for King's and Shop scenes from the Cambridge dataset on

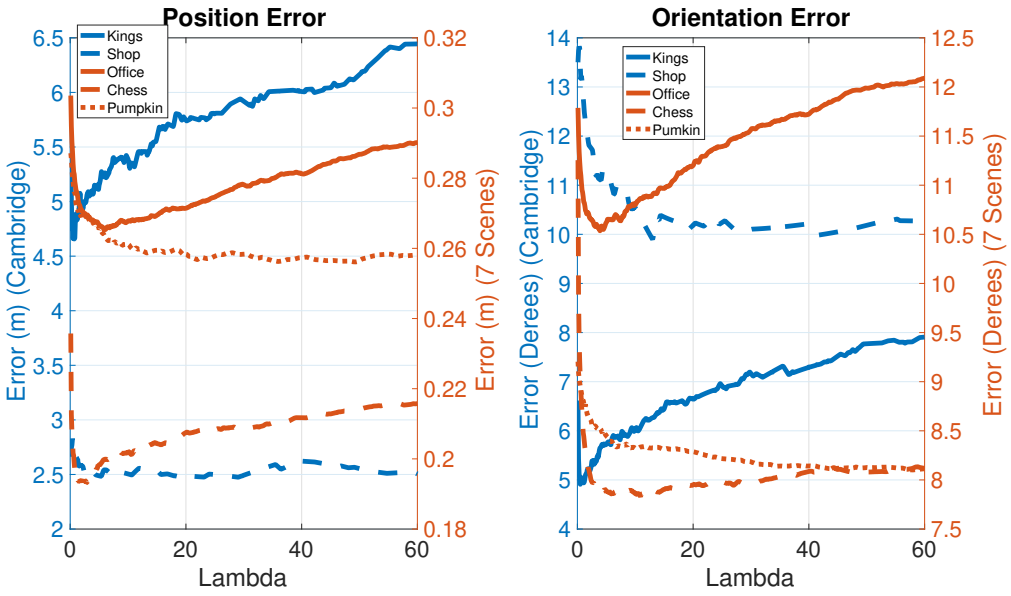


Figure 5.3: Effect of regularization multiplier on the position and orientation error of the 7 scenes and Cambridge datasets.

the left y axis (blue) and Office, Chess and pumpkin scenes from the 7 scenes dataset on the right y axis (orange). We can see that for all the scenes for both position and orientation, λ values ranging from 2 to 12 are the best possible candidates for better performance. Searching for the best value of λ in this range will be fast given that training only needs 0.46 seconds on CPU.

5.5.3 The number of principal components for PCA

For more dimensionality reduction of the input image, we use PCA to select the principal components of the pretrained features of the input image. The number of the principal components K is also a hyperparameter to tune for the best performance. In Fig. 5.4, we study the tuning of K for 2 indoor scenes (Heads and Chess) and 2 outdoor scenes (King's and Shop). We can notice that the error begins to saturate after about 50 components for all the scenes for both systems with small drop on the error till we reach around 280 components to reach the lowest error. Specifically,

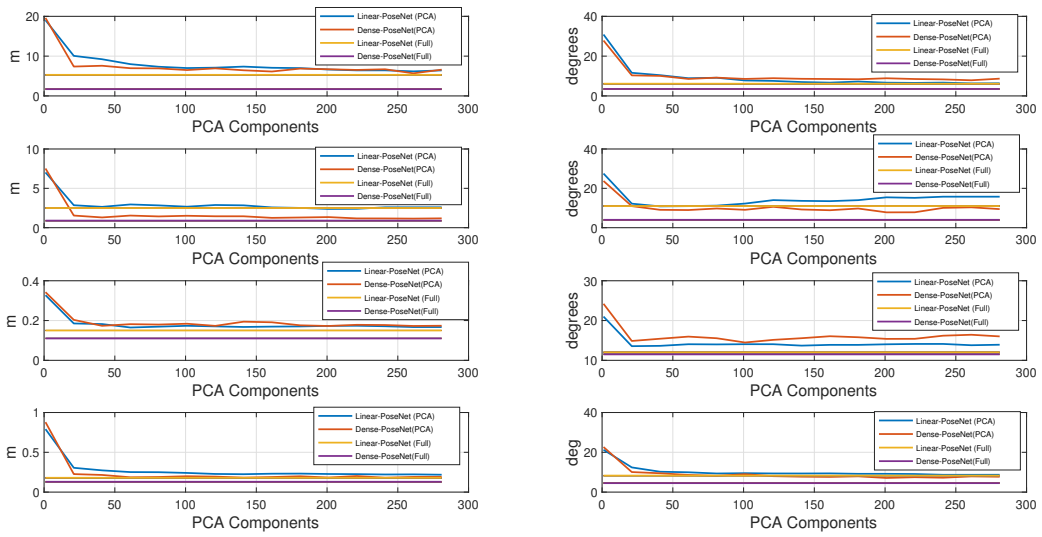


Figure 5.4: The effect of the number of principal components of the pretrained features on the performance of multiple indoor and outdoor scenes compared to using the full set of features.

for Shop scene, we can see that using only 30 PCA components for Dense-PoseNet outperforms the Linear-PoseNet employing the full 2048 features for the position and orientation, leading to a reduction factor of more than 100 in the input size. Moreover, We can use only 30 PCA components for Linear-PoseNet and reach very comparable results to using the full features for Shop and Chess scenes. However for King's and Heads scenes, more components are needed to reach comparable results to the full feature setting for both systems.

5.6 Experimental analysis

We compare the median positional error in meters and median orientation error in degrees of Linear-PoseNet and Dense-PoseNet using both ImageNet and Places dataset with other single image pose estimation methods in Table 5.2 for the 7 scenes and Cambridge dataset. As shown, Linear-PoseNet using ImageNet features reaches

Table 5.2: The median error in position (m)/ orientation (degrees) for the 7 scenes and Cambridge dataset for the proposed models, compared with SurfCNN [12], SURF-LSTM [13], PoseNet [1], G-Posenet [4], Posenet-U [5], Pose-L [14], [6] and G-PoseNet [11], BranchNet [6] ,Mobile-PoseNet [9],VidLoc [7] and Pose-Hourglass [3]

Algorithm	**Chess**	**Fire**	**Heads**	**Office**	**Pumpkin**	**Kitchen**	**Stairs**	**Average**	**Kings**	**St Marry**	**Shop**	**Average**
Linear-PoseNet (ImageNet)	0.18/8.3	0.28/11.5	0.15/12	0.26/10	0.26/8	0.39/12.8	0.42/7.35	0.27/10	5.7/6.15	7.8/18	2.5/11	5.3/11.71
Linear-PoseNet (Places)	0.2/8.3	0.29/13	0.16/11.5	0.27/10	0.28/8.5	0.4/13	0.56/11	0.3/10.7	4.7/4.8	6/14	2.4/9.8	4.34/9.53
Dense-PoseNet (ImageNet)	**0.13/4.5**	**0.22/9.5**	**0.11/11.5**	**0.21**/8.5	**0.23/6.5**	**0.24**/8.5	**0.24/10**	**0.19**/8.4	1.7/3.5	2/6.5	0.9/4	1.46/4.6
Desnse-PoseNet (Places)	0.15/6.9	0.25/8.5	0.15/13	0.28/9.3	0.26/7.5	0.26/9.5	0.27/12	0.23/9.5	1.5/2.9	1.7/6.3	**0.75/3.5**	1.38/4.23
SURF-CNN	0.19/8.10	0.24/12.8	0.17/12	0.35/9	0.36/10.8	0.39/10.2	0.36/10.8	0.3/10.22	7.66/12.3	4.5/10.5	3.2/9.2	6.82/10.6
SURF-LSTM	0.22/7.04	0.24/10.2	0.16/13.6	0.35/8.5	0.35/7.8	0.38/9.5	0.35/11.9	0.3/9.4	5.53/10.6	3.2/9.2	1.46/8.1	5.37/9.3
PoseNet	0.32/8.12	0.47/14.4	0.29/12.0	0.48/8.42	0.47/8.42	0.59/8.64	0.47/13.8	0.44/11.63	1.92/5.40	1.46/8.1	1.11/7.6	2.01/7.03
Dense VLAD	0.21/12.5	0.33/13.8	0.15/14.9	0.28/11.2	0.31/11.3	0.30/12.3	0.25/15.8	0.26/13.11	2.80/5.75	1.11/7.6	1.25/7.5	2.07/6.95
Bay. PN	0.37/7.24	0.43/13.7	0.31/12.0	0.48/**8.04**	0.61/7.08	0.58/**7.54**	0.48/13.1	0.46/9.81	1.74/4.06	1.25/7.5	1.79/6.5	1.70/6.2
PN L-W	0.14/**4.50**	0.27/11.8	0.18/12.1	0.20/5.77	0.25/4.84	0.24/5.52	0.37/10.6	0.24/**7.87**	0.99/1.06	1.05/3.9	1.18/7.4	1.17/4.12
LSTM. PN	0.24/5.77	0.34/11.9	0.21/13.7	0.30/8.08	0.33/7.00	0.37/8.83	0.40/13.7	0.31/9.85	0.88/1.04	1.18/7.4	1.14/5.7	1.19/4.71
G PoseNet	0.20/7.11	0.38/12.3	0.21/13.8	0.28/8.83	0.37/6.94	0.35/8.15	0.37/12.5	0.31/9.94	0.99/3.65	1.14/5.7	3.2/9.2	1.68/6.18
Pose-Hourglass	0.15/6.17	0.27/10.8	0.19/11.6	0.21/8.48	0.25/7.01	0.27/10.2	0.29/12.5	0.23/9.5	NA	NA	NA	NA
Mobile-PoseNet	0.19/8.22	0.37/13.2	0.18/15.5	0.27/8.54	0.34 /8.46	0.31/8.05	0.45/13.6	0.30/10.79	1.14/1.53	2.18/6.1	1.73/6.19	1.6/4.60
Vidloc	0.16	0.21	0.14	0.24	0.36	0.31	0.26	0.25	NA	NA	NA	NA

$0.27m/10^{\circ}$ error which is better or comparable to other methods but with training time less than a second in CPU while other methods require hours of training on GPU. In contract to the good results achieved by Linear-PoseNet for the 7 scenes dataset, the performance is poor for the outdoor scenes compared to other methods with slight enhancement using Places pretrained features. Outdoor scenes are more complex with more details and one layer of linear regression is not enough to learn good features for the pose estimation. However, we solve this issue using Dense-PoseNet which on average outperforms all other methods for the 7 scenes indoor dataset with median error $0.19m/8.4^{\circ}$. For outdoor scenes, Dense-PoseNet reaches comparable results to the other methods using Places pretrained features instead of ImageNet pretrained features with significant reduction in the training complexity. As mentioned in Subsection 5.5.1, we can see the advantages of using Places pretrained features for complex outdoor scenes as lower error is achieved for both position and orientation for all the outdoor scenes.

5.7 Conclusion

We propose Linear-PoseNet that can reach comparable or better accuracy for the single image indoor localization systems with using only one layer of ridge regression and pretrained features of ResNet-50 architecture with training time less than a second on CPU instead of hours of GPU training needed by the state of the art. For outdoor scenes, we show that using only 3 fully connected layers on top of pretrained ResNet50 features without fine-tuning can perform well compared to the state of the art with only minutes of training. For more complexity reduction, we show that downsampling the pretrained ResNet-50 features by more than 10 times using principal component analysis (PCA) has a little effect on the performance but can save both training time and storage space

Chapter 6

Generalizable Sequential Camera Pose Learning Using Surf Enhanced 3D CNN

6.1 Introduction

Learning SLAM systems are based on neural networks that are trained on certain dataset for estimating the pose of the camera. Despite the ability of these systems to mitigate the drawbacks of geometric SLAM systems, they do not generalize well in scenes that are different from the training ones which limit their practical application. Further, the complexity of training procedure is high due to the high dimensional input and networks complexity. Image based localization has been well studied in literature trying to find the absolute pose or the relative pose between two images [93]. However, finding the relative poses between images of video with different frame lengths (video localization) is not well studied despite the great amount of sequential data and features present between the frames of video that can be used to enhance the

localization accuracy. In this chapter, we study the video localization problem trying to mitigate high input dimension (training complexity) by using SURF descriptors of the original image similar to Chapters 2 and 3 and also solve the generalization problems. In particular, we propose two neural networks architectures based on convolutional neural networks (CNN) [33] and recurrent neural networks (RNN) [94]. In one architecture we use the combination of 2D CNN and RNN and the other using 3D CNN only. Further, to reduce the input dimension and the training complexity, instead of using RGB images and pre-trained network such as state of the art methods, we use SURF descriptors [17] which reduces the image dimension by a factor of 48 without the need for a pre-trained network.

Our system accepts a video of N images' SURF descriptors and finds the relative poses (position and orientation) between the first image and the consecutive $(N-1)$ images. The relative pose of the n^{th} frame is represented in the form of a 3D camera relative position Δx_n of size 3 and relative orientation represented by quaternion Δq_n with size 4 as given by Eq. (6.1)

$$\begin{aligned} P_n =& [\Delta x_n, \Delta q_n] \\ \Delta x_n =& \, x_n - x_0 \\ \Delta q_n =& \, q_0^{-1} q_n \end{aligned} \tag{6.1}$$

where x_0, q_0, q_n and x_n are the position and orientation of the first and n^{th} image in a video respectively. Finding the relative poses of all images relative to the first image in the video will help to reduce the drift that can happen if relative poses between consecutive frames are used. Further, it increase the generalization ability of the system.

6.2 Proposed Work

Here, we discuss the components of the current system which takes a video as input, extracts SURF descriptors and then regresses the relative pose between the first image and all the other images in the video using either the combination of CNN and RNN or 3D CNN.

6.2.1 2D CNN-RNN Architecture

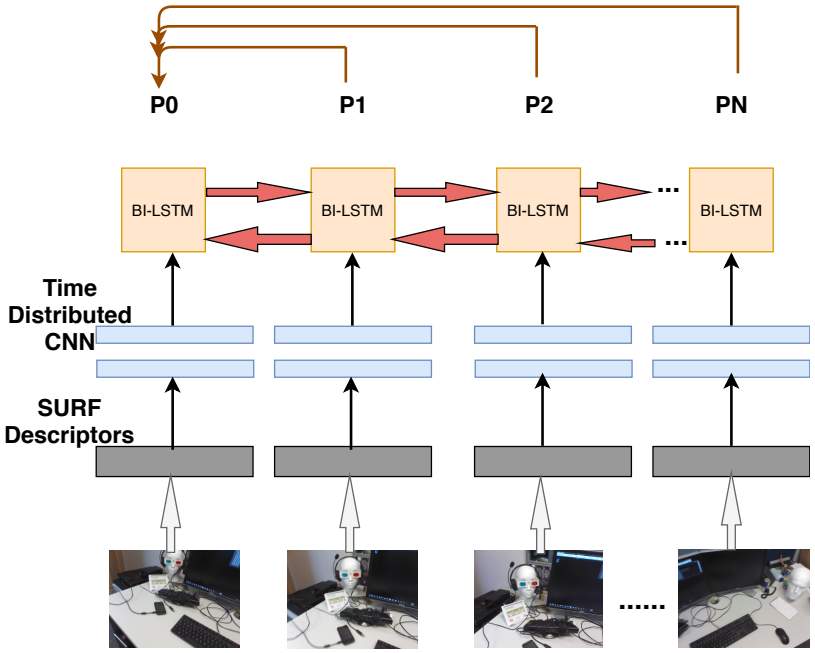


Figure 6.1: 2D CNN-RNN Architecture

In this model, 2D CNNs are used to extract the features from the 2D feature map. As shown in Fig. 6.1, SURF descriptors of all the images of the input video are fed independently to 2D CNN to extract more features and reduce dimension. For finding the relative poses of images relative to the first image, the network needs to learn the temporal features between the extracted features. RNN are well known for its ability to learn from sequential data [94]. Among all RNN types, long short term

memory (LSTM) [56] is very effective in learning sequential features with mitigating the vanishing gradient problem in the vanilla RNN [95]. LSTM consists of input, output, forget and memory gates used to update and learn the sequential features. To learn the sequential relation in both forward and backward direction, we use bidirectional LSTM [18] which consists of the same 4 gates of LSTM but with learning happening from both previous and successive frames which will help learning the relative poses.

Hence, as shown in Fig. 6.1, the network consists of 2 layers of time-distributed 2D CNN across all the images of the input video. The first CNN layer is composed of 64 filter of size 7 x 7 while the second one is 64 filters of 3 x 3. Each CNN layer is followed by RELU activation function, batch-normalization to prevent over-fitting and average pooling layer of size (2 x 2) to reduce the dimensions of the feature maps. Then, bidirectional LSTM with 100 units is used to find the relation between the extracted features. Following that, time-distributed dense layer with 3 neurons for relative position and 4 neurons for relative orientation in quaternion. For the training, the loss function is given by:

$$L_1 = \|\Delta x_t - \Delta x_p\|_2^2 + \left\| \sum_{j=1}^{N} (\Delta x_t(j)) - \sum_{j=1}^{N} (\Delta x_p(j)) \right\|_2^2 + \beta \|\Delta q_t - \Delta q_p\|_2^2 \tag{6.2}$$

where $\Delta x_t, \Delta q_t$ are the ground truth position and orientation of all the images in the video respectively and $\Delta x_p, \Delta q_p$ are the predicted ones. N is the total number of the images in the video. $\Delta x_t(j)$ and $\Delta x_p(j)$ are the true and predicted position of image j in the video. The first and third terms in Eq. (6.2) are Euclidean loss for the relative position and orientation respectively following previous work [1, 7]. We

added the second term to prevent overfitting and make sure that the summation of all the relative positions along the trajectory is correct and avoid the positions to be clustered around the first position which is a sign for overfitting that happened when we used first and third terms only. The hyper-parameter β is used as the position and orientation are of different units. For our calculation, we found that the best value of β is between $200 - 300$.

6.2.2 3D CNN Architecture

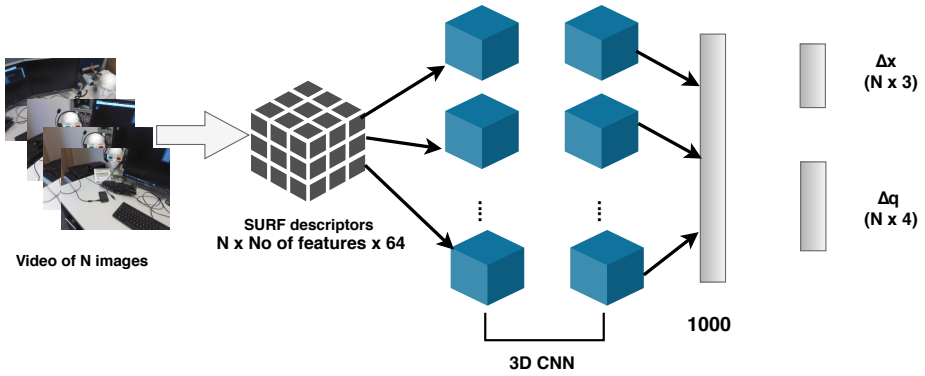


Figure 6.2: 3D CNN Architecture

In addition to 2D CNN-RNN architecture, we propose using 3D CNN to find the relative poses of the images in a video. It is mentioned in the seminal work [96] about using 3D CNN for human activity recognition that 2D CNN-RNN is not as good as the 3D CNN for applications including motion in images. This is because in the case of 2D CNN-RNN, spatial features are extracted independently from all the images and then temporal information is learned using RNN. However, using 3D CNN, the features are extracted in both space and time simultaneously capturing the motion in the video in a better way. As shown in Fig. 6.2, in this model, SURF descriptors of the input images are stacked to form one block with size (number of images, number of features, 64). This block is input to 2 layers of 3D CNN each of which consists of

64 filters with size (3 x 3 x 3) to extract features in both space and time dimensions. Each 3D CNN layer is followed by RELU activation function and average pooling of size (1 x 5 x 5) to reduce the dimensions of the feature cubes. The first dimension is 1 to maintain the number of images of the input to be the same at the output. Next, the extracted features cube is flattened and fed to input to dense layer of 1000 neurons and then the output layer with 3 neurons for the relative position and 4 neurons for the relative orientation. For the loss function we used during training, we found that the Euclidean distance loss function given by (6.3) is sufficient for training without the need to add extra term like the case in 2D CNN-RNN loss function,

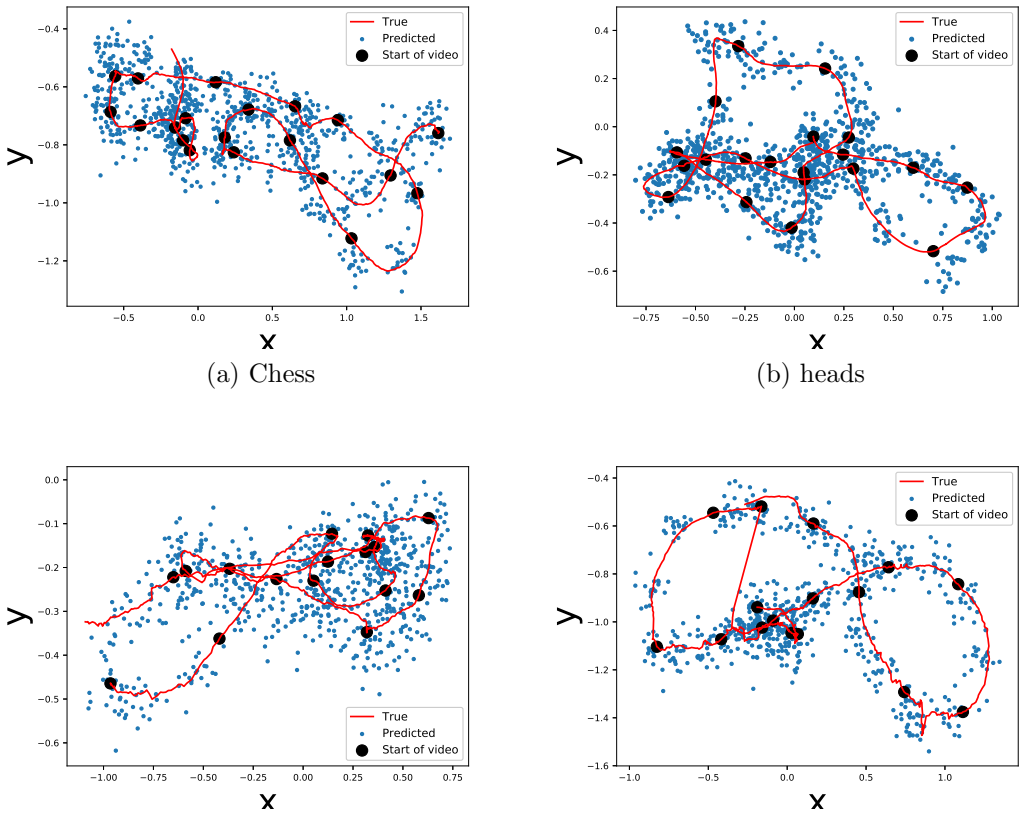
$$L_2 = \|\Delta x_t - \Delta x_p\|_2^2 + \beta \|\Delta q_t - \Delta q_p\|_2^2 \tag{6.3}$$

6.3 Experiments

In our experiments, we use the 7-scenes dataset [57] to validate the proposed systems. In this section, the performance analysis of our two architectures compared with neural networks-based and classical image-based localization systems is discussed. In addition, we compare between 2D CNN-RNN and 3D CNN architectures in terms of accuracy and visualization of learned features.

6.3.1 Generalization

Classical localization systems do not depend on training on certain datasets for finding the poses of images, so they can be generalized to ideally any scene. However, neural network-based localization systems can not be generalized well [3, 7] . Here, our proposed system is designed to mitigate this problem by relating all the poses to the first image in the video. In detail, for the 7 scenes dataset, we train both of



(a) Chess (b) heads

(c) Office (d) Pumpkin

Figure 6.3: Visualization of generalization capability of our system to unknown scenes.

the proposed networks on the heads scene only and test it on all the other different scenes without including any image of them in the training. We show that our system can be generalized to unknown scenes, while all the state of the art work are either trained and tested on the same scene or combine scenes together for training and testing. In Fig. 6.3, the ground truth and predicted trajectories of heads, chess, office and pumpkin scenes with videos of 50 images for the 3D CNN network are plotted with marking the beginning of the video as a black dot. As shown, although the network is trained on the heads scene only, the system is able to predict the positions in the other scenes even for complex environments like the office scene with high accuracy and create trajectories that are consistent with the true ones. The generalization capability of our system will make the learning-based SLAM systems

easier to implement in real time applications as the training is done off-line and testing take only few seconds.

6.3.2 Comparison With Neural Network-Based Systems

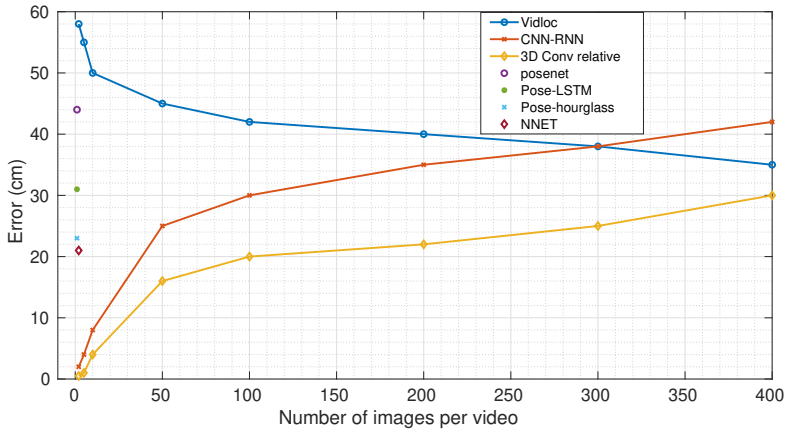


Figure 6.4: The average position error with variable number of imaged per video for the two proposed architectures compared to state of the art

Firstly, we compare the performance of the two proposed architectures with state of the art image-localization work that use neural networks to find the poses. In detail, in Fig. 6.4, we investigate the average position error for the 7 scenes dataset with changing the number of images per video from 2 to 400 images where the current work (2D CNN-RNN and 3D CNN) in orange and yellow curves are compared to video localization method (Vidloc) [7] represented in blue curve as well as single image localization methods including posenet [1], Pose-LSTM [2], Pose-hourglass [3] and NNET [97] represented as dots as they only accept one image as input. As shown, 3D CNN network outperforms the 2D CNN-RNN network, specifically when the number of images increases to be more than 50 images which follows the claim that 3D CNN is more suited for motion-specific applications than CNN-RNN which is

evident with large trajectories. Further, despite that Vidloc's position error is getting lower with increasing the number of images, 3D CNN is outperforming Vidloc for up to 400 images per video and the same case for 2D-CNN RNN system for up to 300 images. Also, our work can be extended to unknown scenes while Vidloc only works for the same training scene as it regresses absolute poses not relative ones which has limitation for use in practical localization and SLAM systems. Finally, taking two images as input, both proposed methods outperform NNET. It is not surprising that posenet, pose-hourglass and pose-LSTM works that accept only one image have higher error than sequential image data localization. The complete analysis for each

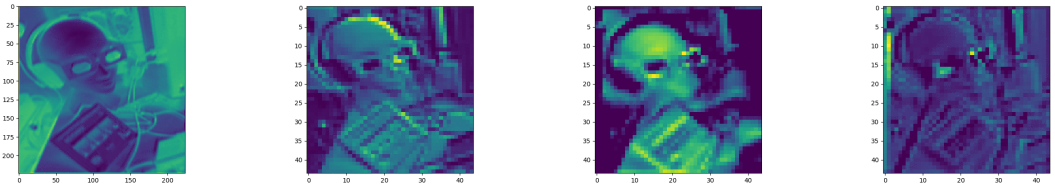
Table 6.1: Comparison of median position and orientation error to the state of the art localization methods.

Scene	**Vidloc**	**Posenet**	**Pose-LSTM**	**Pose-Hourglass**	**NNET**	**Ours(2D CNN-RNN)**	**Ours(3D CNN)**
Heads	0.14m, NA	0.29m, 12	0.21m, 13.7	0.15m, 6.53	0.13m, 6.46	0.187m, 7.26	0.148m, 6.25
Chess	0.18m, NA	0.32m,8.12	0.24m, 5.77	0.15m, 6.53	0.26m, 12.72	0.22m, 8.00	0.16m, 7.28
Fire	0.26 m, NA	0.47m , 14.4	0.34m, 11.9	0.27m, 10.84	0.14m, 12.34	0.195m, 9.89	0.156m, 7.00
Office	0.26 m, NA	0.48m, 7.68	0.30m, 8.08	0.21m, 8.48	0.21m, 7.35	0.25m,10.05	0.201m, 9.02
Red Kitchen	0.31m, NA	0.59m,8.64	0.37m, 8.83	0.27m,10.15	0.24m, 6.35	0.211m , 7.5	0.16m, 6.5
Pumpkin	0.36m,NA	0.47m,8.42	0.37m, 7.00	0.27m,7.01	0.24m, 8.03	0.22m,7.26	0.155m, 8.05
Stairs	0.26m, NA	0.47m,13.8	0.40m, 13.7	0.29m, 12.46	0.27, 11.82	0.197m, 7.67	0.15m, 6.3
Average	0.25m, NA	0.37m, 10.43	0.31m, 9.88	0.23m, 8.85	0.21m, 9.29	0.21m, 8.23	0.164m, 7.2

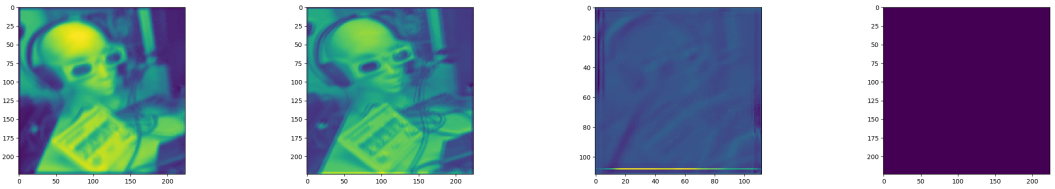
scene in the 7 scenes dataset of median position error and orientation error for videos of 2 to 400 images for both the proposed work and Vidloc along with the median position error for other single and relative image localization systems is detailed in Table 6.1. It is evident from the table that the errors for CNN-RNN and 3D CNN architectures are consistent for all the scenes which prove the capability of our system to work on different environments. Also, the 3D-CNN on average is better than all the state of the art in terms of position and orientation error despite being trained on one scene only while the others do not, which proves the effectiveness of our system.

6.3.3 3D CNN vs 2D-CNN-RNN

Here, to investigate what makes 3D CNN architecture outperform CNN-RNN, we visualize the output of different filters at different layers of both networks. To do so, we train the same networks on raw RGB images instead of SURF descriptors in order to see what each network is learning and what are the features extracted at each layer. Training is done on the heads scene of the 7 scenes dataset. Images of size (480 x 640 x 3) are centered-cropped to be (224 x 224 x 3) and same networks are trained to find the relative poses of images in a video. Outputs of random filters after the convolution and activation of the first and second layer of 2D CNN-RNN and 3D CNN are plotted in Fig. 6.5. As shown, the features extracted from 3D CNN are more distinct than CNN-RNN and features like edges are more obvious in 3D CNN while features from CNN-RNN are not clear to see. Further, starting from the second layer, it is hard to see the output of CNN RNN while features from 3D CNN are very strong. Using this visualization, we can have an insight about the optimality of 3D CNN over CNN-RNN.



(a) 3D CNN



(b) 2D CNN-RNN

Figure 6.5: From left to right: the visualization of the output of $convolutional_1$, $activation_1$, $convolutional_2$ and $activation_2$ of the first and second layers for 2D CNN-RNN and 3D CNN architectures.

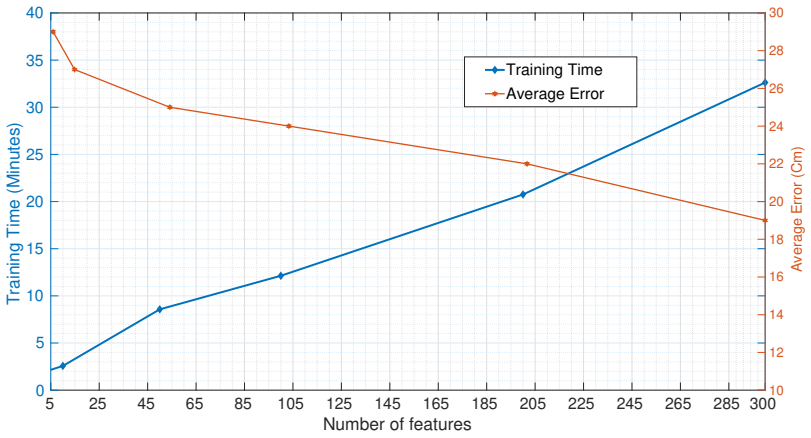


Figure 6.6: The trade off between training time and median position error with changing the number of features per image

6.3.4 Effect Of Varying The Number Of Features

As our system is based on using SURF descriptors instead of RGB images, choosing the number of SURF features to use is very critical. In this subsection, we discuss the effect of varying the number of features of the proposed 3D-CNN system on the error and training time. Training is done on Nvidia TITAN-XP GPU for 30 epochs using ADAM optimizer with learning rate 0.00001. The median position error and training time of videos of 50 images for the heads scene of the 7 scenes dataset are plotted in Fig. 6.6 with changing the number of features per image from 5 to 300 features. As shown, The system can reach 0.29 m error using only 5 features with training in 2 minutes. This will enable us to train for larger videos with lower complexity as the input dimension decreases significantly .The error drops down to 0.19 using the full 300 features with training time equals 32 minutes. Moreover, this plot shows the trade off between training time and complexity, depending on the situation and available computational power. Form this figure, we can choose the best number of features depending on the trade-off between the error and the training time.

6.4 Conclusion

We have proposed two neural network architectures based on 2D CNN-RNN and 3D CNN to find the relative poses of images in videos of different lengths with showing the superiority of 3D CNN for pose estimation. Our system has the ability to generalize well to unknown scenes with simple training by the usage of SURF descriptors of the input images, making the design applicable in real time applications.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this dissertation, we have studied indoor and outdoor localization using camera images. Both single image and video localization using multiple machine learning algorithms are analyzed thoroughly. Our proposed methods can reach good localization accuracy with orders of magnitude reduction in the complexity compared to the state of the art. We conclude from our work with indoor and outdoor scenes that although generally indoor scenes are simpler than outdoor scenes, it is all related to the number of features extracted from each image. In Chapters 2 and 3, we extracted SURF features from the input images before feeding them to the neural networks. These systems only work for indoor environments and failed to generalize to complex outdoor environments as the number of features needed is much higher due to the large field of view and the number of objects in the outdoor scenes. Moreover, for the indoor scenes, the proposed systems work in some scenes much better than the other scenes depending on the number of features in each scene. So it is more related to the complexity of the environment than being indoor or outdoor. To propose sys-

tems that can perform well in complex scenes, we have to use pretrained CNN that is trained on million of images to extract features that represent the image in these complex scenes.

These are the contributions we have achieved.

- In Chapter 2, for a low complexity application, the proposed SurfCNN system utilize the use of SURF descriptors to represent the important features of the input image. A convolutional neural network (CNN) with 5 layers is used to process the SURF descriptors and learn the single image absolute pose in an arbitrary reference frame. The proposed system reaches good localization accuracy with as low as 300 features per image reducing the input dimension by more than 48 times and also reduces the training time to an average of 37 minutes instead of hours needed by other methods.

- In Chapter 3, we extend the idea of using SURF descriptor as a feature representation for single image pose estimation. However, instead of using CNN to process the descriptors, long short term memory (LSTM) network is used to extract the sequential relation between the SURF descriptors sorted from the strongest to the weakest. Using SURF-LSTM, only 50 features are needed to reach similar accuracy to SurfCNN with 300 features and other neural networks based methods. This reduces the training and testing time and storage space of the weights file and image frames.

- In Chapter 4, a single image localization system is proposed using graph neural networks. The features extracted from pre-trained CNN are used for learning the pose which is formulated as graph learning problem. Two approaches are proposed by either representing each image as a node or as a graph. Both of the proposed methods outperform the start of the art for indoor scenes and

produces comparable results for outdoor scene with the superiority of image as a node method over image as a graph architecture.

- In Chapter 5, Linear-PoseNet is proposed which consists of one layer of linear regression for single image pose estimation with the pretrained ResNet-50 features that do as good as or outperforms other methods that use complex neural networks. Linear-PoseNet can be trained in less than a second on CPU which is a huge reduction in the training time compared to other works which needs minutes or hours to train on GPU. Moreover, Dense-PoseNet is introduced with 3 fully connected layers that performs well in outdoor scenes without the need for training large CNN networks. It has been shown that down-sampling the input pretrained features using principal component analysis can make the training faster without sacrificing the performance.

- In Chapter 6, two systems have been proposed based on 3D-CNN and 2D-CNN-RNN for multiple images or video pose estimation that generalize beyond training scenes. The images represented by SURF descriptors are fed to both architectures which are proven to generalize well for multiple indoor scenes. Moreover, the 3D-CNN method outperforms the other architecture as the features are extracted in both space and time simultaneously.

7.2 Future Work

In this section, we discuss the plan for future work beyond this dissertation. In the next subsections, we go through the possible future directions.

7.2.1 Single image pose estimation using RGBD data

In this dissertation, our focus is to use single RGB image for localization and reduce the complexity of these localization systems. Recent state of the art works use the 3D depth data with the RGB data and show very promising results. However, these systems have a very complex training procedure. So the research in the direction of reducing the complexity of these systems can be very fruitful.

7.2.2 Solve the localization problem using other sensors such as LiDAR and WiFi access points

RSSI-CSI Indoor localization using graph neural networks

Using WiFi signals is one of the main topics for indoor localization due to the availability of the devices that can send and receive these signals. WiFi fingerprinting is one of the main approaches for WiFi indoor localization where multiple WiFi fingerprints such as received signal strength indicator (RSSI) and channel state information (CSI) are used to distinguish between different locations [98]. Multiple methods were proposed in the literature for WiFi fingerprinting indoor localization using different approaches including KNN [99], SVM [100], convolutional neural networks (CNN) [101], recurrent neural networks [102] and probabilistic methods [103]. We are planning to propose a novel WiFi fingerprinting system using graph neural neural networks (GNN) using both RSSI and CSI. Specifically, we can formulate the problem as follows:

1. Location as a graph where each location is modeled as a graph consisting of many nodes where each node is one received RSSI-CSI scan.

2. Location as a node where location is represented as a node characterized by the average RSSI-CSI received at this location.

2D Lidar relative pose estimation (Lidar odometery)

We are planning to use ranges information of 2D lidar scans for lidar odometery estimation using recurrent neural networks. We can use bidirectional long short term memory (Bi-LSTM) to model the sequential relation between the consecutive N lidar scan ranges to find the relative translation and orientation between the last two scans. The number of ranges per scans depends on the resolution of the lidar used. We can use multiple lidar sensors with different angular resolution and different frequency

1. 360 degrees LDS lidar sensor [104] which has 360 range readings per scan with 5HZ scan rate.

2. Hokuyo Lidar which has angular resoltion of 0.36 degree with total 1080 ranges readings per scans and scan rate of 25 HZ.

7.2.3 Spatio-Temproal graph neural networks (STGNN) for video pose estimation

In Chapter 4, we propose a single image pose estimation system based on graph neural networks and we show that it can outperform all the other state of the art work for indoor scenes. We can use the same idea but with multiple images or videos by representing each image as a graph and processing the video as multiple graphs and using STGNN to process the graph and extract features spatially between nodes in one graph and sequentially between nodes in the subsequent graphs.

7.2.4 3D semantic mapping using graph neural networks

To have a simultaneous localization and mapping (SLAM) system, the agent should be able to simultaneously localize itself and build the map of the environment. In

this dissertation, we only focus on the localization part and a very plausible future direction is the map generation. Most of the literature fuse Lidar and camera data using CNN with either early, middle or late fusion [105]. They use U-Net [106] like architecture with convolutional and de-convolutional layers. This approach is very expensive as they require to generate very large output dimensions which takes very long time to train. Moreover, working on sparse data structure makes more sense for point cloud data. We can take a different approach by using GNN to solve this problem.

- Each graph can be one down-sampled point cloud fused with camera image frame.
- Each node has long feature composed of the Cartesian coordinates of the point, the intensity of the reflected laser and RGB pixel value corresponding to the point [x,y,z,I,R,G,B].

The edges can be calculated using the Euclidean distance in a certain radius. The GNN will try to learn new set of edges corresponding to the semantics of the input. Moreover, we can learn a label for each node which consists of object class of the point.

Publications

1. **Elmoogy, Ahmed M.**, Xiaodai Dong, Tao Lu, Robert Westendorp, and Kishore Reddy. "SurfCNN: A descriptor enhanced convolutional neural network." In 2019 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), pp. 1-5. IEEE, 2019.

2. **Elmoogy, Ahmed M.**, Xiaodai Dong, Tao Lu, Robert Westendorp, and Kishore Reddy Tarimala. "SurfCNN: A Descriptor Accelerated Convolutional Neural Network for Image-Based Indoor Localization." IEEE Access 8 (2020): 59750-59759.

3. **Elmoogy, Ahmed**, Xiaodai Dong, Tao Lu, Robert Westendorp, and Kishore Reddy. "SURF-LSTM: A Descriptor Enhanced Recurrent Neural Network For Indoor Localization." In The 2020 IEEE 92nd Vehicular Technology Conference: VTC2020-Fall, 2020.

4. **Elmoogy, Ahmed**, Xiaodai Dong, Tao Lu, Robert Westendorp, and Kishore Reddy. "Linear-PoseNet: A Real Time Camera Pose Estimation System Using Linear Regression and Principal Component Analysis." In Multi-Sensor Positioning and Navigation for Connected Autonomous Vehicles Workshop, IEEE VTC2020-Fall, 2020.

5. **Elmoogy, Ahmed**, Xiaodai Dong, Tao Lu, Robert Westendorp, and Kishore Reddy. "Generalizable Sequential Camera Pose Learning Using Surf Enhanced 3D CNN." In Multi-Sensor Positioning and Navigation for Connected Autonomous Vehicles Workshop, IEEE VTC2020-Fall, 2020.

Bibliography

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

[2] F Walch1 C Hazirbas, L Leal-Taixé1 T Sattler, S Hilsenbeck, and D Cremers. Image-based localization using LSTMs for structured feature correlation.

[3] Iaroslav Melekhov, Juha Ylioinas, Juho Kannala, and Esa Rahtu. Image-based localization using hourglass networks. *arXiv preprint arXiv:1703.07971*, 2017.

[4] Ming Cai, Chunhua Shen, and Ian Reid. A hybrid probabilistic model for camera relocalization. In *British Machine Vision Conference*, 2018.

[5] Alex Kendall and Roberto Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *2016 IEEE international conference on Robotics and Automation (ICRA)*, pages 4762–4769. IEEE, 2016.

[6] Jian Wu, Liwei Ma, and Xiaolin Hu. Delving deeper into convolutional neural networks for camera relocalization. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5644–5651. IEEE, 2017.

[7] Ronald Clark, Sen Wang, Andrew Markham, Niki Trigoni, and Hongkai Wen. Vidloc: A deep spatio-temporal model for 6-dof video-clip relocalization. In

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6856–6864, 2017.

[8] Zakaria Laskar, Iaroslav Melekhov, Surya Kalia, and Juho Kannala. Camera relocalization by computing pairwise relative poses using convolutional neural network. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 929–938, 2017.

[9] Claudio Cimarelli, Dario Cazzato, Miguel A Olivares-Mendez, and Holger Voos. Faster visual-based localization with mobile-posenet. In *International Conference on Computer Analysis of Images and Patterns*, pages 219–230. Springer, 2019.

[10] Ahmed M Elmoogy, Xiaodai Dong, Tao Lu, Robert Westendorp, and Kishore Reddy. Surfcnn: A descriptor enhanced convolutional neural network. pages 1–5, 2019.

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

[12] Ahmed M Elmoogy, Xiaodai Dong, Tao Lu, Robert Westendorp, and Kishore Reddy Tarimala. Surfcnn: A descriptor accelerated convolutional neural network for image-based indoor localization. *IEEE Access*, 8:59750–59759, 2020.

[13] A. M. Elmoogy, X. Dong, T. Lu, R. Westendorp, and K. Reddy. Surf-lstm: A descriptor enhanced recurrent neural network for indoor localization. Oct 2020.

[14] Florian Walch, Caner Hazirbas, Laura Leal-Taixe, Torsten Sattler, Sebastian Hilsenbeck, and Daniel Cremers. Image-based localization using lstms for struc-

tured feature correlation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 627–637, 2017.

[15] Jing Wang, Ratan K Ghosh, and Sajal K Das. A survey on sensor localization. *Journal of Control Theory and Applications*, 8(1):2–11, 2010.

[16] Lidar vs. camera — which is the best for self-driving cars? — by vince tabora — 0xmachina — medium. https://medium.com/0xmachina/lidar-vs-camera-which-is-the-best-for-self-driving-cars-9335b684f8d#:~:text=Cameras%20on%20Tesla's%20cars%20use,real%20time%2C%20using%20light%20pulses. (Accessed on 01/27/2021).

[17] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Computer vision and image understanding*.

[18] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[19] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[20] Neil E Klepeis, William C Nelson, Wayne R Ott, John P Robinson, Andy M Tsang, Paul Switzer, Joseph V Behar, Stephen C Hern, and William H Engelmann. The national human activity pattern survey (nhaps): a resource for assessing exposure to environmental pollutants. *Journal of Exposure Science and Environmental Epidemiology*.

[21] Ali Yassin, Youssef Nasser, Mariette Awad, Ahmed Al-Dubai, Ran Liu, Chau Yuen, Ronald Raulefs, and Elias Aboutanios. Recent advances in indoor localization: A survey on theoretical approaches and applications. *IEEE Communications Surveys & Tutorials*, 19(2):1327–1346, 2016.

[22] Shunsuke Miura, Li-Ta Hsu, Feiyu Chen, and Shunsuke Kamijo. GPS error correction with pseudorange evaluation using three-dimensional maps. *IEEE Transactions on Intelligent Transportation Systems*, 16:3104–3115, 2015.

[23] Hui Liu, Houshang Darabi, Pat Banerjee, and Jing Liu. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37:1067–1080, 2007.

[24] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32:1309–1332, 2016.

[25] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and Juan Manuel Rendón-Mancha. Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review*, 43:55–81.

[26] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[27] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision.*

[28] Piotr Dollár, Ron Appel, Serge Belongie, and Pietro Perona. Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36:1532–1545, 2014.

[29] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer, 2010.

[30] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *IEEE international conference on Computer Vision (ICCV)*, pages 2564–2571, 2011.

[31] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. ORB-SLAM: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31:1147–1163, 2015.

[32] Yannis Kalantidis, Giorgos Tolias, Evaggelos Spyrou, Phivos Mylonas, and Yannis Avrithis. Visual image retrieval and localization. In *7th International Workshop on Content-Based Multimedia Indexing, Greece*, 2009.

[33] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361, 1995.

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

[35] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

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

[37] Geng Du, Fei Su, and Anni Cai. Face recognition using SURF features. In *Proc. of SPIE Vol*, volume 7496, pages 749628–1, 2009.

[38] Nikolas Engelhard, Felix Endres, Jürgen Hess, Jürgen Sturm, and Wolfram Burgard. Real-time 3d visual SLAM with a hand-held rgb-d camera. In *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden*, volume 180, pages 1–15, 2011.

[39] Ricardo Chincha and YingLi Tian. Finding objects for blind people based on SURF features. In *IEEE International Conference on Bioinformatics and Biomedicine Workshops (BIBMW)*, pages 526–527, 2011.

[40] Ertugrul Bayraktar and Pinar Boyraz. Analysis of feature detector and descriptor combinations with a localization experiment for various performance metrics. *arXiv preprint arXiv:1710.06232*, 2017.

[41] Lin Chen, Franz Rottensteiner, and Christian Heipke. Invariant descriptor learning using a siamese convolutional neural network. In *XXIII ISPRS Congress, Commission III 3 (2016), Nr. 3*, volume 3, pages 11–18. Göttingen: Copernicus GmbH, 2016.

[42] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *IEEE International Conference on Computer Vision (ICCV)*, pages 118–126, 2016.

[43] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4353–4361, 2015.

[44] John A Hertz. *Introduction to the theory of neural computation*. CRC Press, 2018.

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

[46] Kevin Lin, Jiwen Lu, Chu-Song Chen, and Jie Zhou. Learning compact binary descriptors with unsupervised deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1183–1192, 2016.

[47] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33:1255–1262, 2017.

[48] Richard I Hartley. In defense of the eight-point algorithm. *IEEE Transactions on pattern analysis and machine intelligence*, 19:580–593, 1997.

[49] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[50] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Fast image-based localization using direct 2d-to-3d matching. In *IEEE International Conference on Computer Vision (ICCV),*, pages 667–674, 2011.

[51] Sattler Torsten, Bastian Leibe, and Leif Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *IEEE transactions on pattern analysis and machine intelligence*, 39:1744–1756, 2017.

[52] David Forsyth and Jean Ponce. *Computer vision: a modern approach*. Upper Saddle River, NJ; London: Prentice Hall, 2011.

[53] Torsten Sattler, Michal Havlena, Filip Radenovic, Konrad Schindler, and Marc Pollefeys. Hyperpoints and fine vocabularies for large-scale location recognition.

In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2102–2110, 2015.

[54] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.

[55] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[56] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*.

[57] Ben Glocker, Shahram Izadi, Jamie Shotton, and Antonio Criminisi. Real-time RGB-D camera relocalization. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 173–179, 2013.

[58] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[59] Ankur Handa, Thomas Whelan, John McDonald, and Andrew J Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *IEEE international conference on Robotics and automation (ICRA)*, pages 1524–1531, 2014.

[60] CF Chen and CH Hsiao. Haar wavelet method for solving lumped and distributed-parameter systems. *IEEE Proceedings-Control Theory and Applications*, 144:87–94, 1997.

[61] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[62] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15:1929–1958, 2014.

[63] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016.

[64] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[65] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. pages 298–372, 1999.

[66] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. pages 4104–4113, 2016.

[67] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):0756–777, 2004.

[68] Xiliang Yin, Lin Ma, and Xuezhi Tan. A pclr-gist algorithm for fast image retrieval in visual indoor localization system. pages 1–5, 2018.

[69] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. Inloc: Indoor visual localization with dense matching and view synthesis. pages 7199–7209, 2018.

[70] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. Dsac-differentiable ransac for camera localization. pages 6684–6692, 2017.

[71] Torsten Sattler, Qunjie Zhou, Marc Pollefeys, and Laura Leal-Taixe. Understanding the limitations of cnn-based absolute camera pose regression. pages 3302–3312, 2019.

[72] Yoshua Bengio and Yann LeCun. Scaling learning algorithms towards AI. In *Large Scale Kernel Machines*. MIT Press, 2007.

[73] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[74] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[75] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

[76] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[77] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings*

of the 34th International Conference on Machine Learning-Volume 70, pages 1263–1272. JMLR. org, 2017.

[78] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124, 2017.

[79] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

[80] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

[81] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[82] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609, 2019.

[83] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

[84] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al.

Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[85] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Antonio Torralba, and Aude Oliva. Places: An image database for deep scene understanding. *arXiv preprint arXiv:1610.02055*, 2016.

[86] Keras applications. `https://keras.io/api/applications/`. (Accessed on 01/31/2021).

[87] George AF Seber and Alan J Lee. *Linear regression analysis*, volume 329. John Wiley & Sons, 2012.

[88] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[89] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

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

[91] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.

[92] Fengxi Song, Zhongwei Guo, and Dayong Mei. Feature selection using principal component analysis. 1:27–30, 2010.

[93] Yihong Wu, Fulin Tang, and Heping Li. Image-based camera localization: an overview. *Visual Computing for Industry, Biomedicine, and Art*, 1(1):8, 2018.

[94] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[95] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[96] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013.

[97] Ganesh Iyer, J Krishna Murthy, K Gunshi Gupta, and Liam Paull. Geometric consistency for self-supervised end-to-end visual odometry. *arXiv preprint arXiv:1804.03789*, 2018.

[98] Chaimaa Basri and Ahmed El Khadimi. Survey on indoor localization system and recent advances of wifi fingerprinting technique. In *2016 5th International Conference on Multimedia Computing and Systems (ICMCS)*, pages 253–259. IEEE, 2016.

[99] Minh Tu Hoang, Yizhou Zhu, Brosnan Yuen, Tyler Reese, Xiaodai Dong, Tao Lu, Robert Westendorp, and Michael Xie. A soft range limited k-nearest neighbors algorithm for indoor localization enhancement. *IEEE Sensors Journal*, 18(24):10208–10216, 2018.

[100] Rui Zhou, Xiang Lu, Pengbiao Zhao, and Jiesong Chen. Device-free presence detection and localization with svm and csi fingerprinting. *IEEE Sensors Journal*, 17(23):7990–7999, 2017.

[101] Jin-Woo Jang and Song-Nam Hong. Indoor localization with wifi fingerprinting using convolutional neural network. In *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 753–758. IEEE, 2018.

[102] Minh Tu Hoang, Brosnan Yuen, Xiaodai Dong, Tao Lu, Robert Westendorp, and Kishore Reddy. Recurrent neural networks for accurate rssi indoor localization. *IEEE Internet of Things Journal*, 6(6):10639–10651, 2019.

[103] Minh Tu Hoang, Brosnan Yuen, Xiaodai Dong, Tao Lu, Robert Westendorp, and Kishore Reddy Tarimala. Semi-sequential probabilistic model for indoor localization enhancement. *IEEE Sensors Journal*, 20(11):6160–6169, 2020.

[104] 360 laser distance sensor lds-01 (lidar) - robotis. `http://www.robotis.us/360-laser-distance-sensor-lds-01-lidar/`. (Accessed on 06/29/2020).

[105] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[106] Xiaomeng Li, Hao Chen, Xiaojuan Qi, Qi Dou, Chi-Wing Fu, and Pheng-Ann Heng. H-denseunet: hybrid densely connected unet for liver and tumor segmentation from ct volumes. *IEEE transactions on medical imaging*, 37(12):2663–2674, 2018.