LEARNING APPROACHES FOR SCENE LOCALIZATION AND QUALITY SCENE RECONSTRUCTION

LI CHU TAK

MPhil

The Hong Kong Polytechnic University

2023

The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

Learning Approaches for Scene Localization and Quality Scene Reconstruction

Li Chu Tak

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Philosophy

August 2022

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledge has been made in the text.

_____

LI CHU TAK

_____

Abstract

Vision-based autonomous driving techniques are popular in both academia and industry because of the highly cost-effective commodity cameras with high quality output images and the information richness of images. Global Navigation Satellite Systems (GNSS) is well-known for many real-world ego-localization and other related applications. However, GNSS suffers from reflection and blocking due to dense concrete buildings and tall trees, especially in the densely populated urban areas, like Hong Kong. There are also other solutions using high-level sensors like Lidar, Radar and 360 RGB-D cameras. Nevertheless, these solutions still have their respective limitations and are not widely used in various commercial products. Therefore, various technologies including visual place recognition and reconstruction methods discussed in this thesis will be required for achieving a comprehensive autonomous driving system.

Place recognition or localization is an important element to autonomous driving system. Accurate ego location information is crucial for either removing past accumulated errors or future planning. The challenges lie in the variations in appearance, speeds, lighting environments, perspectives and objects. Therefore, we develop a fast algorithm for place recognition, for which fast tracking with the use of historical information and effective representation of a frame have been comprehensively studied to achieve satisfactory recognition performance and minimize computational cost. We name the use of historical information as a tubing strategy which emphasizes the temporal correlation between consecutive input frames.

We take the advantages of recent deep learning techniques; also remove two main barriers of Convolutional Neural Networks (CNNs) , i.e., heavy computational cost and large amount of labelled data, such that deep learning techniques can be used for efficient place recognition methods. We study lightweight CNN models to offer efficient feature extraction and improve an existing automatic training data generation module by considering more variations in conditions. We further propose a way to adaptively use the historical information to tackle the tasks of unknown initial location and efficient recognition. The proposed methods outperform other state-of-the-art methods in terms of both recognition performance and complexity.

To ensure the quality of the extracted features from images, we also study object removal by means of deep learning-based image inpainting for scene reconstruction. By removing unwanted objects like moving vehicles and pedestrians in images, we can have clean images for place recognition. We propose Deep Generative Inpainting Network (DeepGIN) and inpainting model with Multi-Dilation Fusion Block (MDFB) and auxiliary attention learning branch which seek for a better balance of pixel-wise accuracy and visual quality. We show that our proposed models can handle wild images by testing them on several publicly available datasets, Flickr-Faces-HQ (FFHQ), The Oxford Buildings and Places2 datasets. We demonstrate that our inpainting results can be used in other high-level computer vision tasks such as face verification and semantic segmentation. We believe that the inpainting results can also be used in place recognition.

For future research work, we target at developing a more comprehensive recognition system for which our inpainting models are used as pre-processing module to obtain better input images and our tubing strategy is applied to the post-processing stage to obtain better recognition performance. Apart from combining the techniques discussed in this thesis, we would like to develop an online learning strategy to keep the understanding of a path up to date for further enhancing life-long recognition performance.

List of Publications

International journal papers:

[1] Chu-Tak Li, Wan-Chi Siu, Li-Wen Wang, Zhi-Song Liu, and Daniel Pak-Kong Lun, "General Image Inpainting with Plausible Global Semantics and Visual Quality," *submitted to Pattern Recognition*, 2021.

[2] Chu-Tak Li and Wan-Chi Siu, "Fast Monocular Visual Place Recognition for Non-Uniform Vehicle Speed and Varying Lighting Environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1679-1696, Mar. 2020.

International conference papers:

[1] Chu-Tak Li, Wan-Chi Siu, Zhi-Song Liu, Li-Wen Wang, and Daniel Pak-Kong Lun, "DeepGIN: Deep Generative Inpainting Network for Extreme Image Inpainting," *Proceedings, 16th European Conference on Computer Vision* (*ECCV*), pp.5-22, 23-28 Aug. 2020.

[2] Chu-Tak Li, Wan-Chi Siu and Daniel Pak-Kong Lun, "Vision-based Place Recognition Using ConvNet Features and Temporal Correlation between Consecutive Frames," *Proceedings, 2019 IEEE Intelligent Transportation Systems Conference* (*ITSC*), Auckland, New Zealand, pp.3062-3067, 27-30 Oct. 2019.

[3] Chu-Tak Li, Wan-Chi Siu and Daniel Pak-Kong Lun, "Semi-Supervised Deep Vision-based Localization using Temporal Correlation between Consecutive Frames," *Proceedings, IEEE International Conference on Image Processing* (*ICIP*), Taipei, Taiwan, pp.1985-1989, 22-25 Sept. 2019.

[4] Chu-Tak Li, Wan-Chi Siu and Daniel Pak-Kong Lun, "Boosting the Performance of Scene Recognition via Offline Feature-Shifts and Search Window Weights," *Proceedings, IEEE 23rd International Conference on Digital Signal Processing* (*DSP*), Shanghai, China, pp.1-5, 19-21 Nov. 2018.

[5] Chu-Tak Li and Wan-Chi Siu, "Fast Monocular Vision-based Railway Localization for Situations with Varying Speeds," *Proceedings, Asia-Pacific Signal and Information Processing Association Annual Summit and Conference* (*APSIPA ASC*), Honolulu, USA, pp.2006-2013, 12-15 Nov. 2018.

Acknowledgements

I would like to thank my supervisors, Professor Wan-Chi Siu, and Doctor Pak-Kong Lun Daniel, for their support throughout my study.

Special thanks to my parents for their unconditional love.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

APD……..……...Average Pixel Difference
AUC……..……...Area under Curve
BoW…….. ……Bag-of-Words
BP……. ……..……Back Projection
CA…… ……..……Contextual Attention
CNNs…… ……Convolutional Neural Networks
CV…… ……Computer Vision
FID…… ……Fréchet Inception Distance
FMPR……... ……Fast Monocular Place Recognition
FN……..……False Negative
FP…….. ……..……False Positive
FV……..……Feature Vector
GANs…… ……..……Generative Adversarial Networks
GNSS…… ……..……Global Navigation Satellite Systems
HOG……..... ……...……Histogram of Oriented Gradients
KLD……..……..……Kullback-Leibler Divergence
LCD…….. ……..……Loop Closure Detection
LDB…….. ……Local Difference Binary
LPIPS……...……Learned Perceptual Image Patch Similarity
LRT…….. ……...……Light Rail Transit
MDFB……...……..……Multi-Dilation Fusion Block
MSSA……... ……..……Multi-Scale Self-Attention
NTS…….. ……..……Neural Style Transfer
P……..……..……Precision
PDL…….. ……...……Projected Distribution Loss
R …….. ……..……Recall
RoI…… ……...……Region of Interest
SA……..……..……Self-Attention
SLAM……... ……..…Simultaneous Localization And Mapping
SN……. ……...……Spectral Normalization
SPD…..……..……Spatial Pyramid Dilation
SS…….. ……..……Similarity Score
SSE……... ……...……Sum of Square Error
TP …… ……...……True Positive
VPR…….. ……...…Visual Place Recognition
WCL……. ……..Weighted Confidence Level

Chapter 1 Introduction

1.1 Introduction to Scene Localization

Scene Localization is an important topic to Autonomous Driving. It can be solved by means of visual place recognition and/or image retrieval [1]-[3]. An input query image is compared with all the stored images in a geo-tagged database and the best match pair indicates the location information about the input query image based on the corresponding geo-tag in the form of latitude and longitude. Therefore, visual place recognition can be regarded as an indirect scene localization if all possible queries have been visited and stored in the database with geo-tags.

Visual Place Recognition, also known as Visual Scene Recognition and Loop Closure Detection, is a crucial element to Visual Simultaneous Localization And Mapping (Visual SLAM) [4], [5]. The difficulty of visual place recognition lies on the variations in viewpoints, lighting conditions, and frame capturing interval, etc. Note that the same scene/place could be very or completely different at various time slots. Therefore, robust feature extraction and matching have to be studied to achieve satisfactory recognition performance. In recent decade, a number of visual place recognition approaches have been proposed, for which they take images or frames using standard monocular frontal camera as the input queries. The recognition performance can be enhanced by using consecutive images or frames, i.e., video sequences. It is because we must travel along a path gradually without any sudden jump. Hence, a group of consecutive frames can help to improve the recognition confidence. Note that the core function of visual place recognition is to tell whether the current incoming query frame has been recorded in the database and hence current location information can be obtained in real time. Accurate location information is useful for many autonomous driving related tasks such as visual SLAM and navigation as ego-location information (i.e., instant self location information) can be used to rectify accumulated localization and/or recognition errors.

In the present age, Global Navigation Satellite Systems (GNSS) is well known for ego-localization and is widely used for different real-world applications. Nevertheless, GNSS signal sometimes suffers from reflection and blocking because of dense trees and tall buildings, especially the situations in Hong Kong [4]. There are also other solutions based on other types of sensors like wheel counters, gyroscope, and inertial sensors [5]. However, some of these high-level sensor modules are still in high cost and have their corresponding weaknesses, for example, sensitive to various lighting environments and adverse weather conditions. In this context, standard monocular vision-based algorithms are important to ego-localization because of its cost-effectiveness and a large amount of information from input images. In the literature, many components can be included in the development of monocular visual based localization systems [6]-[15] such as vehicle detection [16], [17], frontal vehicle distance estimation [18], visual odometry [19]-[21], traffic light and sign recognition [22]-[25], and railway or lane detection [26], [27]. Therefore, there could be many components in a comprehensive localization system, and the computational cost of each component should be minimized to ensure an efficient system for real-time performance.

Broadly speaking, visual place recognition can be classified into two groups, namely monocular frame-based and multi-frame based. For monocular frame-based visual place recognition, the most straightforward approach is to compare the current incoming query frame with all the frames stored in the database by means of distance measure such as Euclidean (L2) distance and Cosine distance. The database frame which gives the smallest distance to the current incoming query frame would be the best matched place.. A pre-defined similarity threshold can also be used to further enhance the accuracy of the recognition, i.e., eliminate those best matches with low confidence scores. Such an approach is defined as single nearest neighbor searching and it heavily relies on the discriminative power of the image descriptors. FAB-MAP [28] has been one of the most commonly used appearance-based SLAM systems in

which a Bag-of-(Visual)-Words (BoVW) strategy is employed. For BoW method, salient feature points of frames are first detected and then clustered into a number of groups to construct a visual word dictionary. The visual word dictionary helps to evaluate the similarity between two frames. If two frames contain similar numbers and types of visual words, they would have higher similarity score than others. Nevertheless, Milford and Weth [7] reported that salient point detection can be easily fail especially for extreme changes in appearance and lighting environments. In this context, the discriminative power of the image descriptors is severely degraded and is not enough for maintaining satisfactory recognition performance by using only monocular frame-based approaches. Milford and Wyeth suggested the earliest multi-frame-based method named SeqSLAM [7], which takes sequences of consecutive frames as inputs to enhance the recognition performance. The core idea of SeqSLAM is that the recognition result of each single frame can be regarded as a weak match and a group of weak matches can be a confident match to give convincing recognition result. With the use of multiple frames, SeqSLAM attained 100% precision at around 60% recall rate. Inspired by SeqSLAM, many researchers have started to integrate the temporal correlation between consecutive frames into their proposed algorithms [29]-[34]. The temporal correlation here can be regarded as the information given by the previous incoming query frames. For example, if there is no difference or just small difference from the image descriptors of previous frames, a stop ego-motion can be deduced as the frontal camera takes very similar incoming frames continuously.

In the recent decade, the boom of Convolutional Neural Networks (CNNs) has shown that CNNs have outperformed most conventional methods in different computer vision (CV) tasks [35]. The features extracted by CNNs called CNN or deep features, have demonstrated better and robust discriminative power than traditional hand-engineered features. Since 2012, AlexNet [35] has been the first CNN-based method which outperforms conventional methods

in object classification task. For AlexNet, there are five convolutional layers, and each level of convolution can be regarded as certain feature extraction, from concrete to abstract levels. The first level of convolutional layer (conv1) targets at simple feature patterns such as edges, corners, and colors. The fifth convolutional layer (conv5) looks for more complex feature patterns like a dog face, a wheel, and a gun shape. Sünderhauf *et al.* [13] studied the recognition performance of CNN features given by AlexNet in place recognition tasks, for which the skeleton features of a place are the most important. We can know that there can be dynamic and static objects in a place and these objects can be regarded as noises distracting our decision making. Therefore, the middle convolutional layer, i.e., conv3, seems to be more suitable for place recognition tasks as skeleton features are tended to be extracted at this layer from experiments and the skeleton features provide the gist of a frame. Their experimental results haven demonstrated that conv3 features offer the best recognition performance compared to features from other layers. Nevertheless, the dimensionality of the conv3 features is very high in terms of the length of the feature descriptors. For example, for a 224 × 224 frame, the descriptors at conv3 are 384 × 13 × 13 = 64,896. This causes slow pair matching especially for the single nearest neighbor searching or linear full searching. In addition, a large amount of class-labelled images for demanding training can be expensive to different practical applications.

From the success of using deep features, many researchers have started to apply CNNs to visual place recognition [13], [36]-[39]. It has been well known that high computational cost and difficulty in collecting large amount of labelled data are two main weaknesses of CNN-based place recognition methods. Merrill and Huang [36] tried to resolve these two limitations by means of employing a shallow convolution autoencoder network and training the network using self-transformed images. With the shallow fast feature extraction stage and the needlessness of labelled data, they proposed a lightweight and unsupervised place recognition

method. Autoencoder network can be used to extract gist features because of its denoising property [40]. General speaking, there are several convolutional layers to squeeze the dimensionality of the input and extract representative features of the input. We then reconstruct the input image with the squeezed features. In this process, the network has to learn how to extract salient features of the input and reconstruct a "clean" version of the input as output based on the extracted features. This is a straightforward denoising network setting. In [36], they generated the training data by applying random perspective transformation to every image so that each single image is paired up with its transformed version as a training pair. As a result, the trained autoencoder is robust to one practical situation, i.e., changes in perspective.

For visual place recognition nowadays, CNNs always act as a feature extraction stage for extracting representative features of the input frame because of its robustness to variations in changes in lighting conditions, viewpoints, and appearance. Once we have more discriminative and robust feature representation of the input frame, we can further improve the recognition performance. However, discriminative features have to be extracted by deeper CNNs, for which these CNNs are still demanding for real-world applications with limited resources. The field of visual place recognition is still finding an effective and efficient algorithm with fast feature extraction and feature matching. This implies that researchers should work on i) efficient feature extraction with better discriminative power of features and/or ii) efficient feature matching and searching regardless of the size of the database.

1.2 Introduction to Quality Scene Reconstruction

Quality Scene Reconstruction has been an overwhelming topic for Autonomous Driving. It is useful for providing better visual experience of the current capturing scene to users. Apart from enhancing the visual quality of a scene, scene reconstruction can also benefit visual place recognition by means of providing clean inputs without unwanted dynamic objects for recognition. In this work, we consider object removal as a technique for quality scene

reconstruction. For the applications of unwanted dynamic object removal, image inpainting, also known as image completion, is a highly related task to be studied.

Image inpainting is a task of predicting the values of missing pixels in a masked or corrupted image such that the completed image looks realistic and is semantically close to the reference ground truth even though it does not exist in real-world situations. It would be useful for erasing unwanted parts from photos.

With the recent success of deep learning in image recognition, generation, super-resolution, image synthesis and many others [51], [54]-[60], a growing number of CNN-based inpainting approaches [61]-[76] have been proposed to fill in masked images with plausible global semantics in an end-to-end manner. Most of these methods adopt the framework of Generative Adversarial Networks (GANs) [77] because of its capability of generating visually appealing images. This conforms to image inpainting for which we have to generate the missing content with good visual quality. For example, dilated convolution [67] can be used to enlarge the receptive field of a layer without increasing parameters by adjusting the dilation rate such that we can see the context of images with the use of a shallower fully convolutional network. Most of the state-of-the-art inpainting models [62], [64], [68], [70], [74]-[76] follow a two-stage approach which contains a coarse reconstruction stage and then a refinement stage. The first coarse generator roughly fills in the masked images with correct spatial structures, and then the following refinement generator explicitly mends and decorates the generated content with fine details. Yu *et al.* [68] proposed the idea of contextual attention (CA) layer for further improving the textures of the generated content by borrowing information from correlated feature patches at distant spatial locations. Apart from that, feature perceptual and style losses for style transfer and super-resolution to generate high-quality images [58], [59] have also been applied to image inpainting [61], [69], [73], [75] for encouraging similar high-level feature representation and style between the completed images and real images. With certain

understanding of the context, appropriate feature references, and relevant perceptual loss functions, one can reconstruct the missing content with both plausible visual quality and global semantics.

1.3 Organization of the thesis

This thesis is organized as follows. In Chapter 2, we review the basic knowledge of visual place recognition, including types of methods, evaluation metrics, common datasets and challenging in place recognition. We also review the task of image inpainting and focus on the main function of it which is used for dynamic object removal for quality scene reconstruction.

Chapter 3 covers the details of our proposed monocular vision-based place recognition method using both conventional machine learning and recent deep learning techniques. We demonstrate how temporal information helps to improve the recognition performance. Chapter 4 proposes a deep generative inpainting model for quality image reconstruction. We show how a better balance of pixel-wise reconstruction accuracy and visual quality can be found. Lastly, Chapter 5 gives our conclusion of this study.

Chapter 2 Technical Review

2.1 Visual Place Recognition

Visual Place Recognition (*VPR*), Scene Recognition and Loop Closure Detection (*LCD*), generally refer to the same task. *VPR* and *LCD* are the commonly used terms in Robotics society. For "Loop Closure Detection", it is widely used when the travelling path forms a close loop. Scene Recognition, *VPR* and *LCD* share the identical final objective which is reporting the fact that the current capturing scene/place has been previously visited or not.

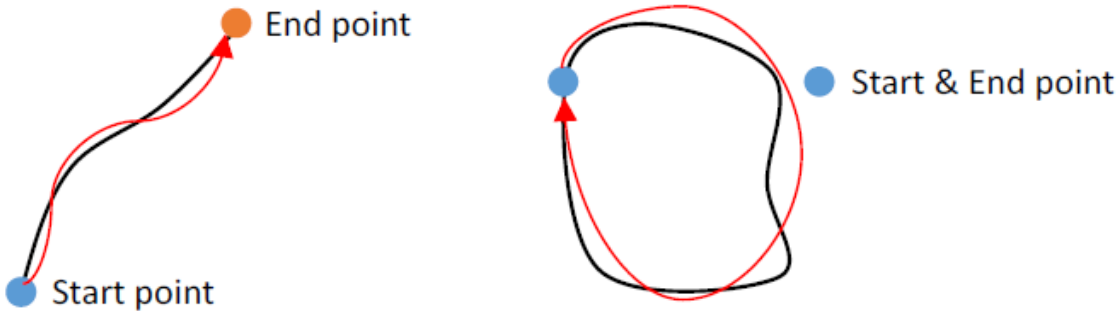


Figure 2.1 Illustration of the visual place recognition tasks

Figure 2.1 is an illustration of different types of paths that we usually study in visual place recognition and loop closure detection. The first kind of path is shown on the left-hand side. It is a path from the start point to the end point without any closing. The second kind of path is located at the right-hand side, for which it has the same start and end point. Hence, this path forms a close loop, and we detect a loop closure when we re-visit the start point.

Figure 2.2 Some examples of place recognition and/or loop closure detection

Figure 2.2 shows some examples of place recognition and/or loop closure detection. It is observed that the agent, i.e., the robot or the vehicle, re-visits the sample place under changes in conditions like changes in lighting environments and perspective views.

2.1.1 Types of Visual Place Recognition

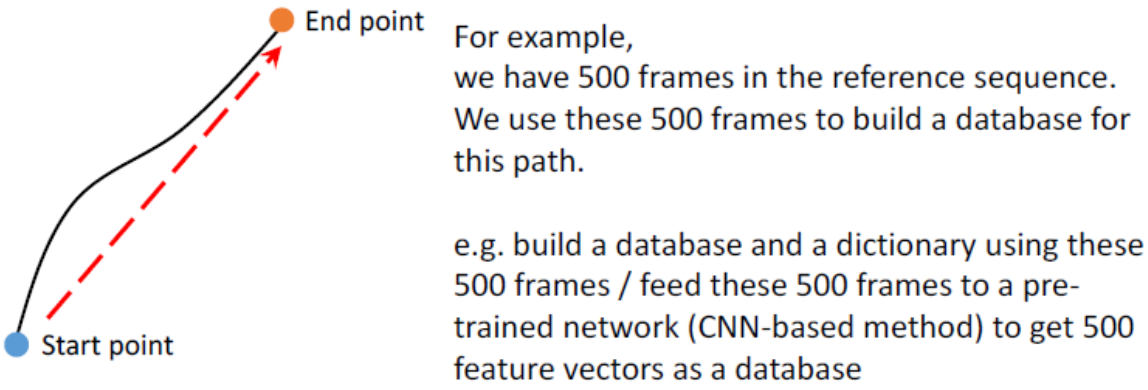


Figure 2.3 An example to show a standard visual place recognition approach

For standard visual place recognition methods, we usually have an offline learning stage to store the path in terms of a number of frames first. This implies that we have to learn the path from a reference sequence and there would not be any update once the path is learned. In this context, we would have a pre-defined database. We then compare each input frame with all the frames stored in the database to give recognition or loop closure detection result. Figure

2.3 shows an example of a standard visual place recognition approach. For example, we have 500 frames in the reference sequence and these 500 frames would be used to build a database for the corresponding path. Of course, there are many ways of building the database. We may input the 500 frames to a pre-trained CNN to get 500 feature descriptors or use conventional feature extraction methods to extract features from the 500 frames to form the database.

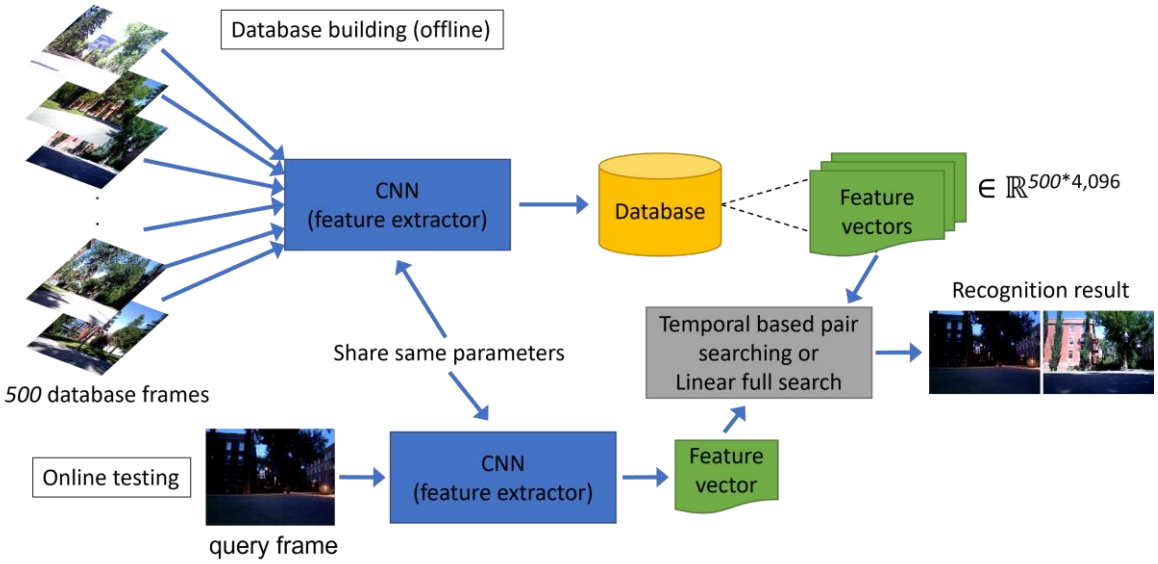


Figure 2.4 An example of standard building of a database and standard visual place recognition testing procedure with the fixed database

Figure 2.4 shows an example of how we use a pre-trained CNN to build a database of a reference sequence. Assume that we want to use a pre-trained VGG16 [41] as the feature extractor to build the database. Note that the final fully connected layer of the VGG16 is with the feature length of 4,096. We can feed the 500 frames to the VGG16 and get 500 feature vectors with the length of 4,096. As a result, we have 500 feature vectors, one vector represents one scene/place. We then store all the 500 feature vectors to the database and obtain a recognition database with 500 places. Note also that the database would not be changed unless we rebuild the entire database.

After building the database, we can perform online recognition. Assume that we now have an incoming frame, and we would like to match this frame to all the frames in our database. The first step is to feed the new scene frame, i.e., query frame, to the pre-trained VGG16 to get the feature vector (FV) with the length of 4,096. We then compare this FV with all the FVs stored in the database. More specifically, we compare the FV with FV001, FV002, …, FV500 (i.e., feature vectors of database frame 1, 2, …, 500) using distance measure techniques such as Euclidean (L2) distance and Cosine distance. The match pair with the smallest distance would be the best match and it implies the recognition result. A pre-defined threshold can be used to judge whether the best match represents the same place or not. If the distance between the FV of the query frame and the FV of the 9^{th} database frame is the smallest and the distance is smaller than a pre-defined threshold, FV009 will be the final recognition result. This means that the new incoming scene is matched to scene 9 in the database, and they represent the same place.

2.1.2 Evaluation Metrics

2.1.2.1 Precision-Recall Curve

Current frame	Reported match	Ground truth	Correct/Incorrect (Y/N)
1	3	1	N
2	2	2	Y
3	3	3	Y
4	4	4	Y
5	5	5	Y
6	-	6	-
7	-	7	-
8	8	8	Y
9	9	9	Y
10	9	10	N
Summary: tp = 6 (6‘Y’); fp = 2 (2‘N’); fn = 2 (2‘-’) Precision = 6 / (6 + 2) = 0.75 Recall = 6 / (6 + 2) = 0.75			

Precision (***P***) [0, 1]	Recall (***R***) [0, 1]	*F*1
0.8	0.8	0.8
0.8	0.5	0.62
1.0	0.5	0.67
0.5	1.0	0.67
0.9	0.9	0.9
1.0	0.7	0.82

Table 2.1 Evaluation metrics – Precision-Recall Curve (PR Curve) and F1 Score with numerical examples

For visual place recognition and loop closure detection, one of the commonly used evaluation metrics is precision-recall curve. Precision (*P*) is defined as the ratio of true positive reported recognition results (*tp*) to the total reported recognition results (*tp*+*fp*) while Recall (*R*) is defined as the ratio of true positive reported recognition results (*tp*) to the total ground truth recognition results (*tp*+*fn*). Table 2.1 gives a simple numerical example of how the precision and recall are calculated. In this case, we have 10 frames, and the approach gives 8 confident matches. Note that there are 2 frames we cannot confidently report the matches, hence the false negative, *fn* = 2. Also, among the 8 reported matches, 6 of them are correct.

Therefore, we get *tp* = 6 and *fp* = 2 and finally the precision and recall of this example are both 0.75 as shown in the example.

2.1.2.2 Area Under Curve and Max. Recall Rate at 100% Precision

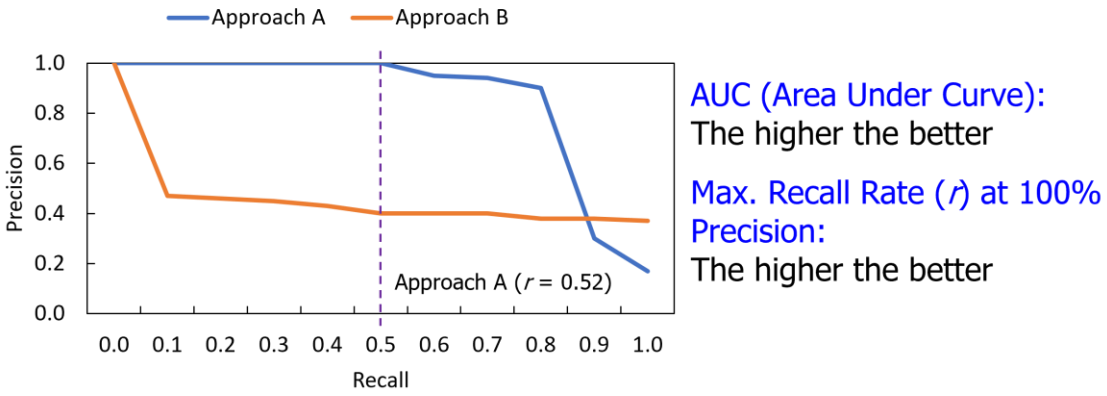


Figure 2.5 Evaluation metrics – Precision-Recall Curve (PR Curve) and Area Under Curve (AUC), an example for visualization

Figure 2.5 shows an example of Precision-Recall curve (*PR* curve) that we found in the visual place recognition or loop closure detection paper [36]. We have two main ways to explain the *PR* curve. Area Under Curve (AUC) is the first interpretation, the higher the better. The second way is the maximum recall rate (*r*) at 100% precision, also the higher the better. In the robotics community, the loop closure detection results are usually used to eliminate accumulated errors for mapping rectification. Hence, robotics researchers target at a perfect precision even at lower recall rate. They look for the maximum recall rate at 100% precision. In the *PR* curve of Figure 2.5, we can see that the maximum recall rate at 100% precision of the deep blue curve is around 0.52.

2.1.2.3 F1 Score

Apart from Area Under Curve (*AUC*) and Max. recall rate at 100% precision, Table 2.1 also shows another commonly used metric in visual place recognition, named F1 Score. In this example, we show a simple numerical example of how to compute the F1 Score. Practically, we target at both high precision and recall. In order to get high F1 Score, we have to get both high precision and recall. The F1 Score is calculated as,

$$F1 = 2 \times \frac{P \cdot R}{P + R} \tag{2.1}$$

where P is the precision and R is the recall rate. For high $F1$, both P and R have to be high.

2.1.3 Datasets for Visual Place Recognition

Datasets	Number of sequences	Total number of frames in average	Time-synchronized	Remarks
LRT-S	4	623	No	Changes in speeds, lightings, and motion blur
LRT-L	4	2566	No	
UA [43]	2	646	Yes	Day-night sequences
Nordland [42]	4	5950	Yes	Seasonal changes
Alderley [7]	2	2035	Yes	Extreme changes in weather and lightings

Table 2. 2 Table summarising all the datasets used

Table 3.2 summarises all the datasets used for our study of visual place recognition. We included sequences with different lengths and various practical issues encountered in real life. Note that some sequences have been time-synchronized such that frames with same indices represent the same places.

2.1.3.1 Nordland and UA Datasets

Figure 2.6 A typical dataset for scene recognition – Nordland dataset [42]

Figure 2.6 shows one of the widely used datasets for visual place recognition, named Nordland dataset [42]. There is no loop closure in this dataset. One important point is that the Nordland dataset has been time-synchronized so as to focus on the extreme changes in appearance and seasons.

Figure 2.7 More examples of both Nordland dataset [42] and UA dataset [43]

Figure 2.7 gives more example of the Nordland and UA datasets [42], [43]. The UA dataset focuses on the changes in lighting conditions. In practical situations, we need to handle variations in lighting conditions and appearance, motion blurring, seasonal changes, as well as varying speeds.

2.1.3.2 Looping Datasets

Figure 2.8 Some examples of loop closure datasets [10], [28].

Figure 2.8 displays some examples of loop closure detection datasets [10], [28]. For this kind of looping datasets, the vehicle or the car-like robot would travel the loop using a number of branches. For example, the map at the right-hand side in Figure 2.8. there are two

purple loops and we can only perform the place recognition in the overlapping area of the two loops (the red circle). In this context, we may consider updating the database in an online manner in order to recognize all the visited places. This is related to the topic of online place recognition and is not covered in this thesis.

2.1.4 Challenging in Visual Place Recognition

In this part, we summarize the challenging in visual place recognition into two areas. The first area is the extreme changes in conditions. It is difficult to have a sufficient generalization on the extracted deep features for frame matching as incoming frames suffer from variations in seasonal changes and blurring, changes in viewpoints, lighting environments, and appearance, and different speeds.

The second area is related to the real-time performance commitment. Visual place recognition is similar to large-scale image retrieval tasks, but we have to pay attention to the complexity of the entire system. When the processing time per frame is longer than the frontal camera capturing interval, i.e., the frame rate, a delay in our location information would be introduced.

2.1.5 Recent Development of Visual Place Recognition

Recently, Chancán and Milford [98] further developed SeqSLAM [7] and proposed DeepSeqSLAM. Similar to our proposed methods in this thesis, they took the advantages of both deep features and sequential information from historical frames. They used a Convolutional Neural Network (CNN) model to extract deep features from incoming frames and a Recurrent Neural Network (RNN) model to integrate temporal information over a sequence of consecutive frames. Their work re-confirms the use of CNN models for place recognition and the importance of temporal information to place recognition. They showed that the use of temporal information can further reduce the false positive rates compared to the

single nearest neighbour approaches. We show that the idea of using temporal information has been covered in our work.

Apart from this, Patch-NetVLAD [99] has been proposed as an improved version of NetVLAD [2], [3]. They employed the idea of multi-scale fusion of locally global feature descriptors to take the advantages of both local and global descriptors. Such an approach can effectively handle both appearance and viewpoint changes. Note that local keypoint and patch descriptors are effective against perspective changes while global descriptors are effective against changes in appearance. This idea is similar to how we define key frames and represent the key frames in our work. We analyse key frames by using conventional machine learning techniques such that we can use few but effective local feature patches with variable sizes to represent a key frame. This reflects that the idea of conventional salient points or features has been integrated into recent deep learning-based feature representation methods for better representation of the input images.

2.2 Image Inpainting

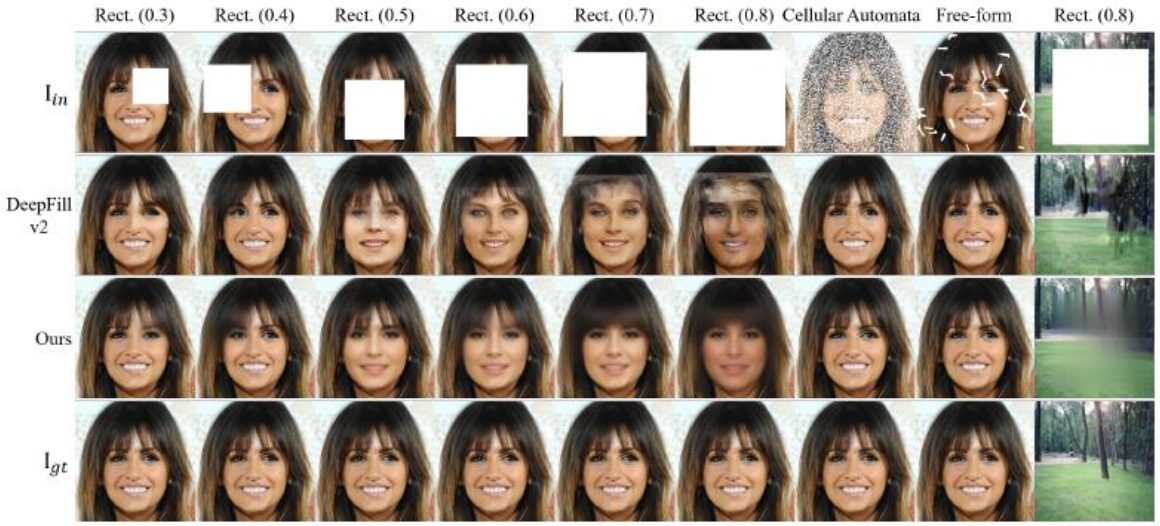


Figure 2.9 Degree of difficulty in extreme image inpainting

Image inpainting is a task of filling in the missing areas in a masked image such that the completed image is a prediction of the original image with similar appearances, semantics, textures, and details. The degree of difficulty depends highly on the scales and forms of the missing regions as well as the contents of both the valid and invalid pixels as shown in Figure 2.9. The first row shows the input masked images $\mathbf{I}_{in}$ with the corresponding mask described on top of them. Rect. (α) represents a random rectangular mask with the height and width rate of α of each dimension. The randomly generated mask based on cellular automata is introduced in the AIM 2020 Extreme Image Inpainting Challenge [78], and the free-form mask is proposed in DeepFillv2 [62]. The last row displays the ground truth images.

2.2.1 Problem Formulation

Let us formulate the image inpainting problem with the help of Figure 2.10. We first define an input RGB masked image and a binary mask image as $\mathbf{I}_{in} \in \mathbb{R}^{H\times W\times 3}$ and $\mathbf{M} \in \mathbb{R}^{H\times W}$ respectively. The pixel values input to the model are normalized between 0 and 1 and pixels with value 1 in $\mathbf{M}$ represent the masked regions. $\mathbf{I}_{coarse} \in \mathbb{R}^{H\times W\times 3}$ denotes the output of the coarse reconstruction stage (light orange). We also define the output of the refinement stage (light green) and the reference ground truth as $\mathbf{I}_{out} \in \mathbb{R}^{H\times W\times 3}$ and $\mathbf{I}_{gt} \in \mathbb{R}^{H\times W\times 3}$ respectively. Note that H and W are the height and width of an input/output image, and we fix the input to 256 × 256 for inpainting. Our objective is to complete $\mathbf{I}_{in}$ conditioned on $\mathbf{M}$ and produce a completed image $\mathbf{I}_{out}$ ($\mathbf{I}_{compltd}$) which should be both visually and semantically close to the reference ground truth $\mathbf{I}_{gt}$. $\mathbf{I}_{compltd}$ is the same as $\mathbf{I}_{out}$ except the valid pixels are directly replaced with the ground truth. The network is trained under the framework of generative adversarial learning with training data $\{\mathbf{I}_{in}, \mathbf{M}, \mathbf{I}_{gt}\}$ where $\mathbf{M}$ is randomly generated with arbitrary sizes and shapes. The coarse generator takes $\mathbf{I}_{in}$ and $\mathbf{M}$ as input to generate $\mathbf{I}_{coarse}$ as output. Then, $\mathbf{I}_{coarse}$ and $\mathbf{M}$ are fed to the refinement generator to obtain the completed image $\mathbf{I}_{out}$ ($\mathbf{I}_{compltd}$).

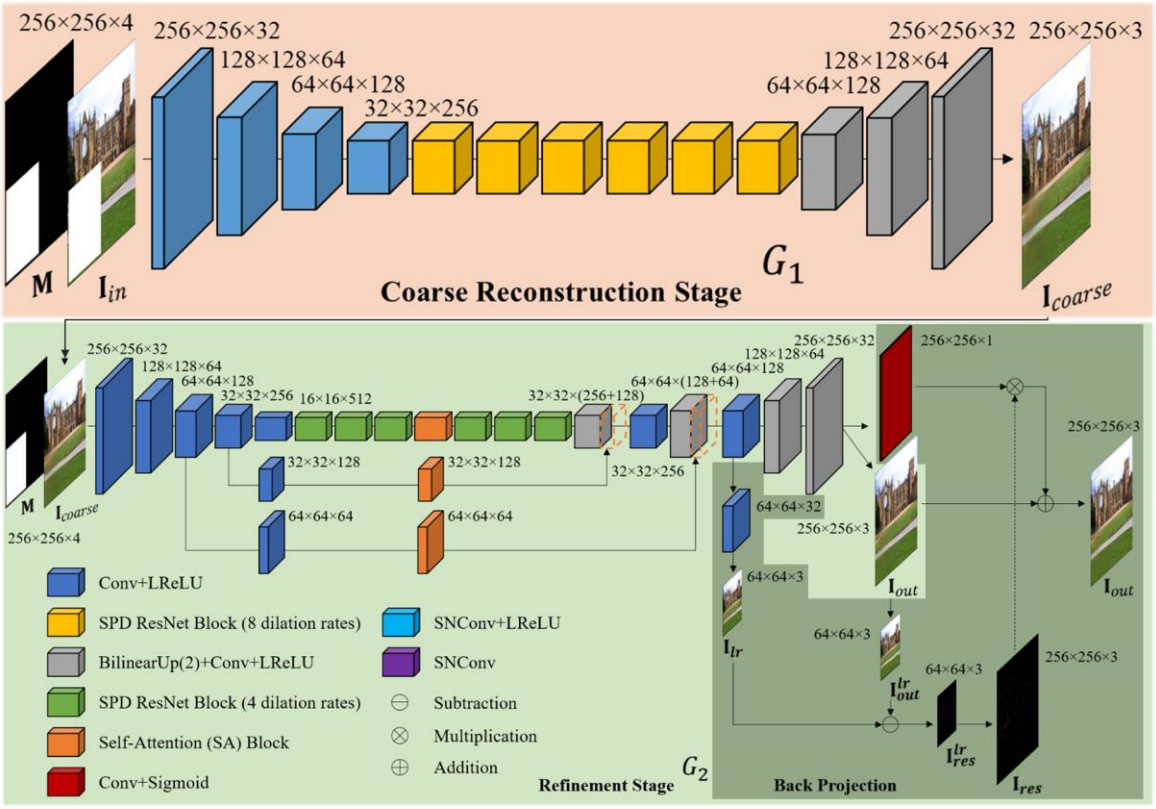


Figure 2.10 Architecture of the generators of our DeepGIN [75]

2.2.2 Types of Random Masks

2.2.2.1 Regular Mask

Pathak *et al.* [65] proposed the first deep learning based inpainting algorithm that employs the framework of Generative Adversarial Networks (GANs) [77] for more realistic image completion. They resized images to 128 × 128 and assume a 64 × 64 rectangular centre missing region for the task of inpainting. The encoded feature of the image with the centre hole is then decoded to reconstruct a 64 × 64 image for the centre hole. Random rectangular masks with different height and width rates are shown in the first six columns of Figure 2.9 for reference.

2.2.2.2 Irregular Mask

For the early stage of deep learning-based methods of image inpainting, authors focused on the rectangular types of masks and this assumption limits the effectiveness of these methods

in real-world situations. Liu *et al.* [61] addressed this problem by suggesting a partial convolutional layer, for which a binary mask for indicating the missing regions is automatically updated along with the convolutional operations inside their model for guiding the reconstruction. Yu *et al.* [62] further improved the concept of partial convolution by proposing gated convolution for free-form image inpainting. An example of the proposed random free-form masks can be seen at the second last column of Figure 2.9.

Another type of masks has been introduced in the AIM 2020 Image Inpainting Challenge [78], in which the masks are randomly generated based on cellular automata. An example of this type of masks is provided at the seventh column of Figure 2.9.

Note also that object masks are also valid in the case of object removal using different image inpainting models.

2.2.3 Common Techniques for Image Inpainting

2.2.3.1 Generative Adversarial Networks

Context Encoder [65] is the first deep learning-based inpainting algorithm that adopts the framework of Generative Adversarial Networks (GANs) [77]. For GAN-based image inpainting, a generator is designed for filling the missing regions with semantic awareness and a discriminator is responsible for distinguishing the completed image and the reference ground truth. Under this framework of GANs, the generator and discriminator are alternately optimized to compete against each other, as a result, the completed image given by the generator would be visually and semantically close to the reference ground truth. Based on this early work, most if not all later developed inpainting models are also trained under the framework of GANs.

2.2.3.2 Dilated Convolution

For the task of image inpainting, understanding the context of images is crucial for filling the missing regions with plausible global semantic. Pathak *et al.* [65] proposed channel-

wise fully connected layer for which all feature locations at the previous layer contribute to each feature location at the current layer. Nevertheless, fully connected layer limits input images to fixed size or requires additional layers to control the spatial sizes with a model, hence multiple models are usually required for images with different resolutions. To release this limitation, Iizuka *et al.* [67] presented a fully convolutional network with dilated convolutions which allows us to understand the context without using the channel-wise fully connected layer. More specifically, for a 3 × 3 filter with dilation rate = 1, its receptive field is 3 × 3 and we have 9 learnable parameters. If we would like to enlarge the receptive field to 5 × 5 such that each feature location at the current layer can see more of its neighbours, we may use a 5 × 5 filter with dilation = 1 which results in 25 learnable parameters. To achieving the desired receptive field without inducing dozens of parameters, alternatively we can choose a 3 × 3 filter with dilation rate = 2. It remains 9 parameters, while its receptive field is 5 × 5 (= 3 + 2 × (*d*-1), where *d* is the dilation rate = 2 in this example). This idea forms the basis of all state-of-the-art models [61]-[64], [68]-[76]. For the recent ECCV 2020 AIM challenge on extreme image inpainting [78], DMFN [73] and DeepGIN [75] extended this idea to multi-dilation convolutional blocks and achieved outstanding performance in terms of PSNR, i.e., pixel-wise reconstruction accuracy.

2.2.3.3 Gated Convolution

Early CNN-based inpainting models [65]-[68] focused on regular masks. This limits the practicability of these models to real-world inpainting problems with arbitrary masks. Liu *et al.* [61] suggested partial convolution for which a binary mask is simultaneously updated with the convolutional operation to indicate the reduced missing regions for the guidance on generating the missing content. Yu *et al.* [62] enhanced their previous model [68] by proposing a learnable version of partial convolution, called gated convolution. The hard-assigned binary mask in partial convolution was modified to a soft-gated convolutional layer which gives much

flexibility for indicating the validness of each feature location. The soft gate can be implemented by an additional branch of a convolutional layer followed by a sigmoid activation.

2.2.3.4 Contextual Attention

Recent state-of-the-art inpainting models [62], [64], [68], [74]-[76] utilize the concept of "patch matching then replacing" to improve local fine details of the generated content by borrowing appropriate reference feature patches from the known regions. For example, Yu *et al.* [62] proposed a contextual attention (CA) layer which reconstructs the generated feature patches inside the missing regions via weighted sums of all the extracted reference feature patches, and the weights are derived from the similarities between the generated and the reference feature patches. To remove the heavy loading of calculating the similarity between each pair of patches and ensure the appropriateness of the reference feature patches to each generated feature patch, Zeng *et al.* [76] proposed to perform the CA operation only during training by means of a contextual reconstruction loss. An additional branch of reconstructing target images using information only from the similarity between each pair of patches was designed for optimizing a per-pixel accuracy ($L1$) loss and an adversarial loss. As a result, the correctness of the nearest reference feature patches can be enhanced, and better reference feature patches can be attached to the main branch for the missing content generation.

2.2.3.5 Perceptual Losses

With the advent of neural style transfer (NTS) [58], [59], some researchers have employed relevant losses, e.g., style loss and feature perceptual loss [61], [66], [69]-[75] for the task of image inpainting. Their core idea is to transfer the style of the extracted reference feature patches to the generated feature patches through the minimization of one or more perceptual losses. For example, Yang *et al.* [66] improved the textures of the results obtained from Context Encoders [65] through proposing a texture network with a local texture loss. The texture network is a well pre-trained VGG19 network [41] on ImageNet [49] for image

classification. For the local texture loss, they extract the features at the middle layers as computed by the texture network and then encourage each generated feature patch to have similar feature responses at its nearest reference feature patch by minimizing their L2 distance. There are also other inpainting methods [61], [69]-[75] trained with the variants of the style and/or perceptual loss to generate realistic textures. A recent study on perceptual loss [79] has found that L1 or L2 distances between extracted features and distances between distributions of extracted features, e.g., Kullback-Leibler Divergence (KLD), are not necessarily appropriate for producing perceptually realistic results. They proposed to aggregate 1D Wasserstein distances between each pair of individual feature maps from input and target, computed by any well pre-trained CNN model. The proposed loss is called projected distribution loss (PDL).

2.2.4 Dataset for Image Inpainting

2.2.4.1 ADE20K Dataset

A subset of the ADE20K dataset [80], [81] was selected as the train set for the AIM extreme general image inpainting challenge [78] of the ECCV 2020. This dataset is collected for scene parsing and understanding, in which it contains images from various scene categories. The subset was provided by the organizers of the challenge, and it consists of 10,330 training images with diverse resolution roughly, from 256 × 256 to 3648 × 2736. Figure 2.11 shows some examples of this dataset.

Figure 2.11 Examples of the ADE20K dataset [80], [81]

2.2.4.2 CelebA-HQ Dataset

Figure 2.12 Examples of the CelebA-HQ dataset [82]

Apart from the ADE20K dataset, the CelebA-HQ dataset [82] is also commonly used for facial image inpainting. Figure 2.12 shows some examples of this dataset. This dataset contains 30,000 high-quality facial images with a standard size of 1024^2. Some previous methods like [69], [73], [75] randomly split this dataset into two groups, 27,000 images for training a face image inpainting models and 3,000 images for testing.

2.2.5 Challenging in Image Inpainting

As mentioned in section 2.2 and Figure 2.9, the difficulty of image inpainting highly depends on the sizes of the missing area and the semantic importance of the missing regions to the entire image. For example, if half of a person or a car is removed, it is more difficult to recover the whole person or car because of its semantic significance. In this case, we still have half of the person or car in the masked image and only limited information is available to help to fill in the missing regions. In contrast, it is easier to fill in the missing regions if the entire person or car is masked. We can fill in the missing regions with the background information, like the sky and grass field. Therefore, we have to seek a better balance of the semantic correctness and the visual quality. Completed images with correct global semantic are important to other high-level tasks such as semantic segmentation and image recognition.

The second challenge is related to the network design of inpainting models. The necessity of the two-stage approach has been challenged by some one-stage inpainting models [61], [65]-[67], [69], [71]-[73] which are also able to produce state-of-the-art high-quality completion results. Computationally expensive Contextual Attention (CA) layer has been recently restudied [76] to improve its effectiveness and efficiency by replacing the generated feature patches by the selected reference feature patches at a later layer which shares the same spatial size as the input to preserve more local textures and eliminating its expensive search operation from inference stage. Hence, inpainting network design is another interesting topic, in which the effectiveness of each building block should be studied.

For the most recent development of deep learning-based image inpainting, Zhou *et al.* [100] proposed TransFill which is a reference-guided image inpainting model. They fill in the hole by referring to a warped reference image with learned spatial transformations. As a result, they can offer inpainting results with plausible visual quality compared to other state-of-the-art inpainting models without using any reference image. Apart from reference-based image

inpainting, high-resolution and large mask image inpainting tasks are also hot research and industrial topics. Zeng *et al.* [101] proposed Aggregated Contextual Transformation GAN (AOT-GAN) for high-resolution image inpainting. They designed an AOT block, for which multi-dilation convolutional layers are used to include various receptive fields for capturing both near and distant spatial neighbours. Their experimental results showed that the proposed AOT block brings impressive improvements by capturing both near and distant spatial locations for high-resolution images and large missing regions. Suvorov *et al.* [102] also proposed a new model named Large Mask inpainting, LaMa. They designed their inpainting network using fast Fourier Convolutions (FFCs), which offers a receptive field that covers the entire image, and it is less sensitive to the input image size. For existing inpainting models, more layers are required such that a model can see the entire image when the input resolution is large. With FFCs, a model can capture the global context of an image at early layers and hence high-resolution images with large missing regions can be completed.

In this thesis, we also demonstrate that the range of the receptive fields of an inpainting model is important to the inpainting performance. We show that self-attention can be used to enhance the visual quality of the inpainting results by borrowing the high-frequency textures from the valid pixels to refine the generated pixels. Apart from these, we focus on seeking the balance of visual quality and pixel-wise reconstruction accuracy such that the inpainting can be further used for other high-level computer vision tasks like verification, segmentation and recognition.

Chapter 3 Learning Approaches for Place Recognition

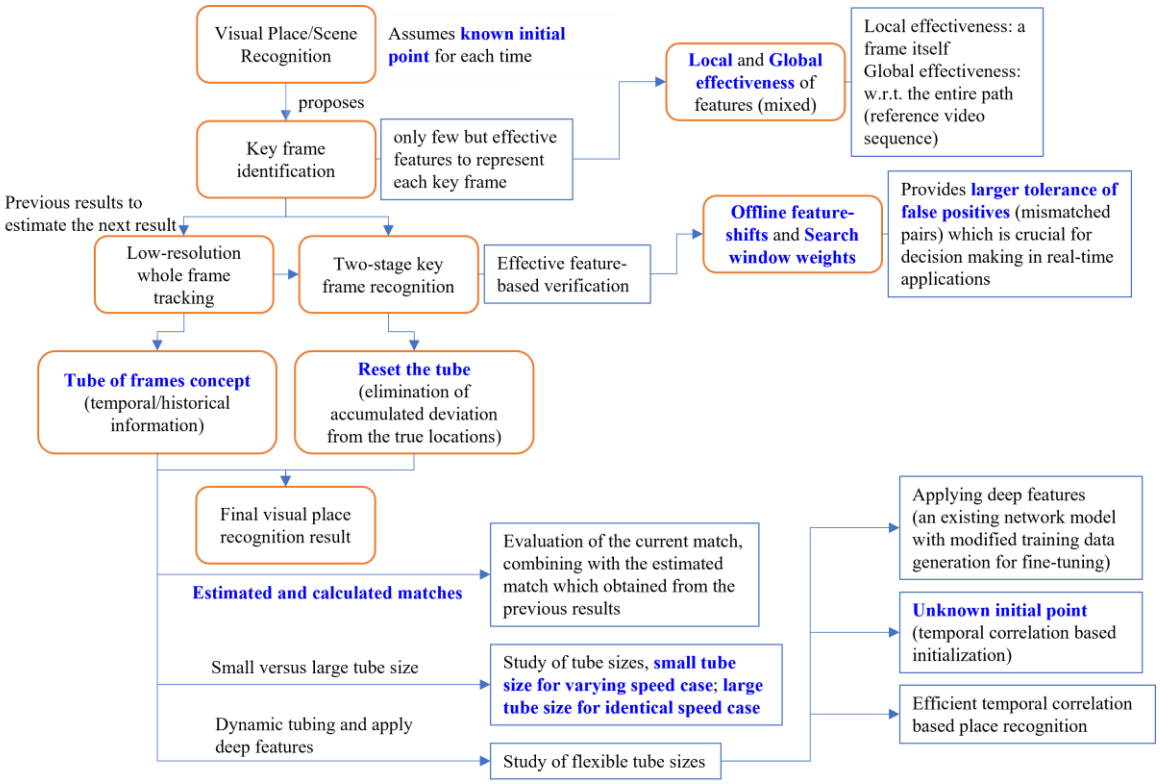


Figure 3. 1 Overview of the block diagram of the work done on Visual Place Recognition. Our study started with known initial point. We assume that some frames are easier to be recognized (key frames) compared to other frames and few but effective features can be used to represent the key frames. By utilizing the fact that a vehicle must travel along a path gradually, we propose the tube of frames concept to take the previous recognition results into account.

Figure 3.1 shows the block diagram of our workflow on visual place recognition. The orange blocks are the core components in our algorithm. At the beginning, we assume that the initial point of a path is known. We identify the key frames in a path by means of studying the local and global effectiveness of the features in a frame. We then use low-resolution whole frame tracking with the concept of tube of frames for fast reporting of the recognition results. We also design a two-stage key frame recognition to eliminate the accumulated efficient tracking errors. The offline feature-shifts and search window weights are also proposed to consider the practical feature-shifts in real time for the key frame recognition. This is our conventional machine learning approach for place recognition.

To extend the concept of tube of frames, we also study dynamic tubing strategy and the use of deep features for place recognition. We propose a deep learning-based method for tackling the task of unknown starting point of a journey. We also apply the same idea to achieve efficient temporal correlation-based place recognition.

In this chapter, we first dive into each component of our conventional machine learning approach for visual place recognition. We then introduce how we extend our ideas to deep learning-based efficient method of visual place recognition.

3.1 Machine Learning Approach for Place Recognition

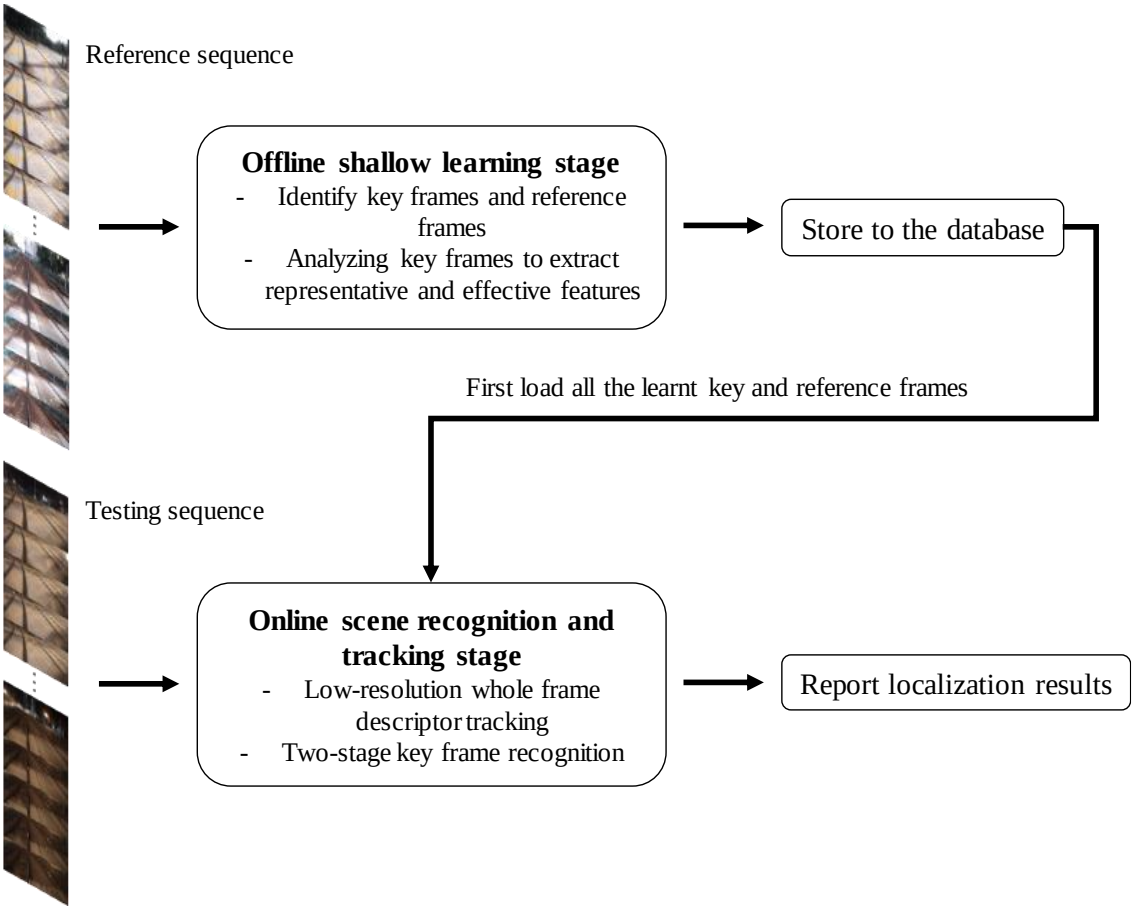


Figure 3. 2 Overview of offline shallow learning stage and online scene recognition and tracking stage for Fast Monocular Visual Place Recognition (FMPR)

Figure 3.2. provides an overview of our proposed Fast Monocular Visual Place Recognition (FMPR) method. We design an offline shallow learning stage to learn a path from

a reference sequence. Key frames and reference frames are identified and stored for online recognition and tracking stage. A few but effective features of the key frames are extracted for better representative power of the key frames to enhance the recognition accuracy and efficiency. With these two strategies, we can build a database for the path. For our proposed online recognition and tracking stage, also named as online practicing stage later in this chapter. We propose a low-resolution whole frame descriptor tracking approach, for which recognition results are given by an effective comparison with neighbouring frames and a prediction of current location based on previous recognition results. With the proposed tracking, the efficiency of the system can be improved. We also design a two-stage key frame recognition to rectify the accumulated errors from fast tracking. The “two-stage” arrangement is designed in which fine recognition process will only be activated when more information is needed for decision making. In the following, we first introduce the proposed offline shallow learning stage in which key frames and reference frames are defined and extracted. Then, we describe the details of the online practicing stage.

3.1.1 Offline Shallow Learning Stage

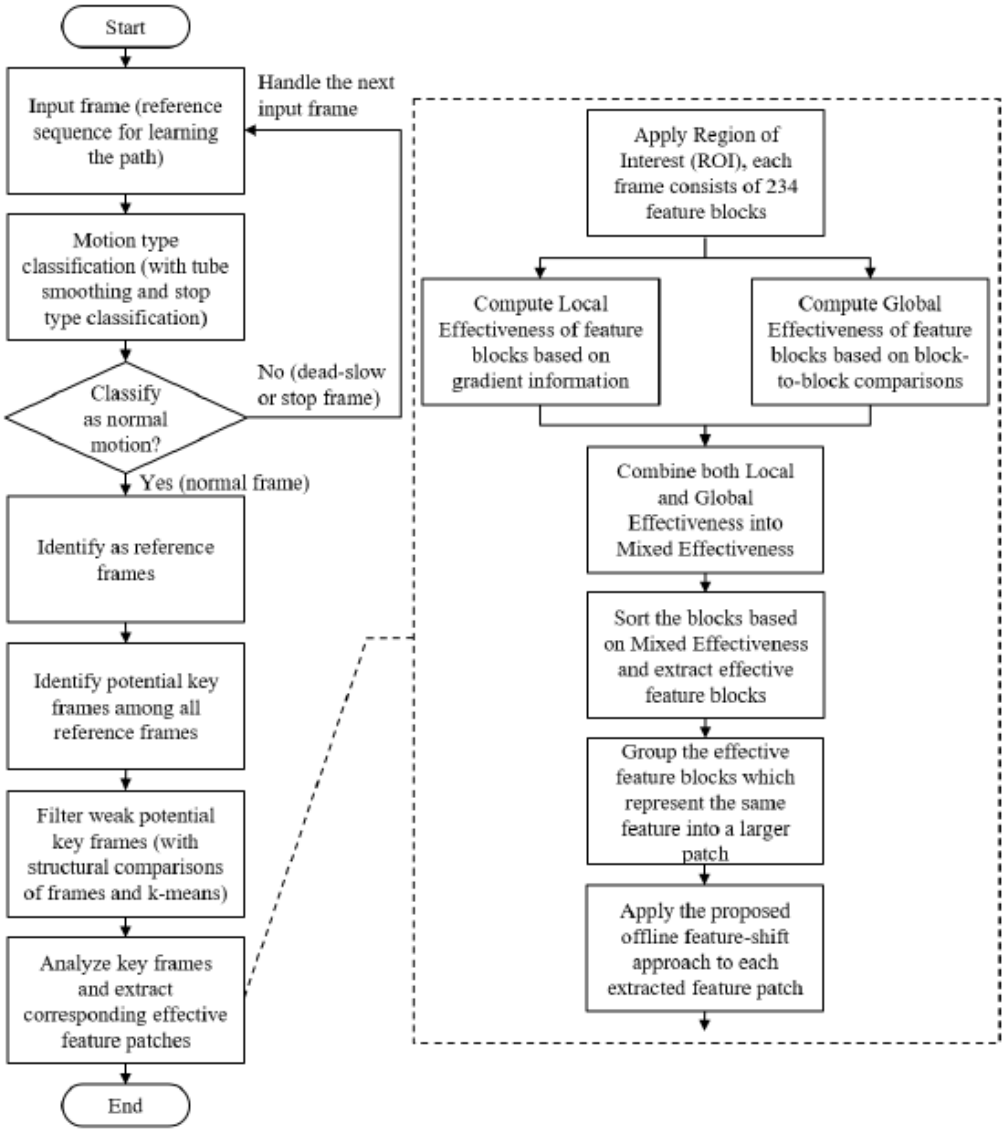


Figure 3. 3 Flowchart of the proposed offline shallow learning stage

Figure 3.3 displays the flowchart of our offline shallow learning stage. In this stage, we target at learning a path by means of extracting representative information from a reference sequence and storing the information in the database for online usage. The box drawn with the dash line shows how the analysis of key frames is performance. We learn the path from the reference sequence in an offline manner. We start to introduce the motion types of the reference sequence as displayed in the 2nd box of the left-hand side of Figure 3.3. All the other components in Figure 3.3 are fully discussed in the following sub-sections.

3.1.1.1 Motion Type Classification

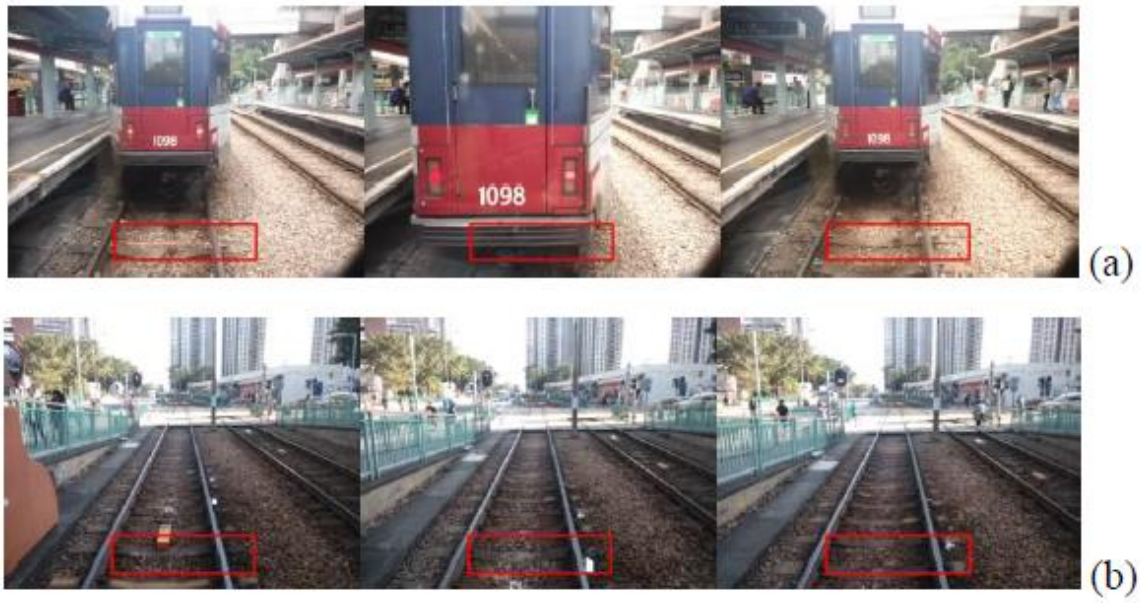


Figure 3. 4 (a) First type of stop motions: Vehicle in the front. (b) Second type of stop motions: No vehicle in the front

The core idea of the motion type classification is to detect duplicated incoming frames with stop or dead-slow motions. Milford and Wyeth [7] focused on handling of identical speed and constant motion situations which are not sufficiently practical for real-world applications. Therefore, we start our study on stop and dead-slow motions, in which redundant or even confusing information can be identified. More importantly, removing repeated frames can save the database storage and speed up the processing time. There are two types of stop frames in our case studies. Figure 3.4(a) shows the first type of the stop motions which encounters a vehicle in the front. A stop frame offset is designed to eliminate the effect of the motion of the frontal vehicle, or otherwise the motion of the frontal vehicle plays an important role in visual based methods instead of the self-motion we are studying. The second type of the stop motions is also shown in Figure 3.4(b) and there is no frontal vehicle. Hence, there is no need to apply the stop frame offset to this type of stop motions.

One effective way to capture the types of stop motions is to compute the average pixel difference within the red-bounded region of two consecutive frames as shown in Figure 3.4. The size of the red-bounded region is defined as 256 × 64 pixels. We down-sample the region to 64 × 16 and divide it into a number of 8 × 8 small regions for contrast normalization so as to minimize the effect of local changes in illumination. A search window is constructed for capturing the dead-slow motion of the vehicle and the average pixel difference are computed as:

$$d(\Delta m, \Delta n, I_i, I_{i-1}) = \frac{\sum_{n=0}^{H-1}\sum_{m=0}^{W-1}|I_i(m+\Delta m, n+\Delta n) - I_{i-1}(m,n)|}{W \times H}, i \in [1, F-1] \quad (3.1)$$

where W and H are the width and height of the down-sampled normalized red-bounded region, 64 and 16 respectively. $I_i(m, n)$ is the intensity value of the pixel located at (m, n) of the down-sampled region of frame i. F is the total number of reference frames in the studying reference sequence. Δm and Δn are the shifts in terms of pixels. We shift the location of the region according to Δm and Δn before extracting and down-sampling the red-bounded region. We look for $\Delta m_{\min}$ and $\Delta n_{\min}$ that minimize the average pixel difference, d(.), between two consecutive frames.

$$(\Delta m_{\min}, \Delta n_{\min}) = \underset{\Delta m, \Delta n \in [-s, s]}{\arg\min}\ d(\Delta m, \Delta n, I_i, I_{i-1}) \quad (3.2)$$

where s is the search range in terms of pixels and the minimized $d(.)$ for frame i, d_i, is the final average pixel difference value between frame i and i-1.

$$d_i = d(\Delta m_{\min}, \Delta n_{\min}, I_i, I_{i-1}) \quad (3.3)$$

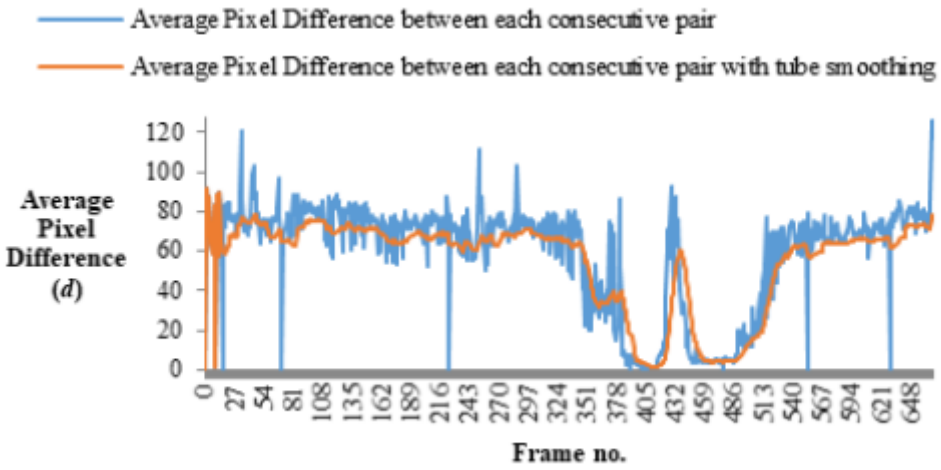


Figure 3. 5 Average pixel difference between each consecutive pair with (orange) and without (blue) tube smoothing

To deal with exceptional cases like repeated frames during capturing the current place and variations in the average pixel difference values as shown in the blue curve of Figure 3.5. We adopt "tube smoothing", $d_{i,\text{tube}}$, which computes the average pixel difference of the current frame by averaging the "average pixel differences" of the neighbouring frames to smooth out the exceptional cases, please refer to the orange curve in Figure 3.5. If $d_{i,\text{tube}}$ is smaller than the pre-defined thresholds, $T_{\text{dead-slow}}$ and T_{stop}, the motion type of the current incoming frame is classified as dead-slow and stop motion respectively.

3.1.1.2 Reference Frame Identification

Apart from the dead-slow and stop frames, all the other frames are identified as reference frames with standard motion in the proposed method. Note that all the reference frames would be stored in the database in the form of feature descriptors for online practicing stage.

3.1.1.3 Potential Key Frame Identification

Potential key frames are identified from the reference frames, and we define key frames as frames that can be represented by few discriminative features with respect to the entire

reference sequence. We assume that key frames can confidently be recognized with only few but effective representative features. We define a reference term, *AVG* $d_{\text{ref frames}}$, i.e., the average of the "average pixel differences" among the reference frames. The *AVG* $d_{\text{ref frames}}$ is computed and rounded up as follows:

$$\text{ROUNDUP}\left(AVG\ d_{\text{ref frames}} = \frac{\sum_{r=0}^{R} d_r}{R}\right) \tag{3.4}$$

where R is the number of reference frames and d_r is the final average pixel difference of the reference frame r defined in Eq. 3.3.

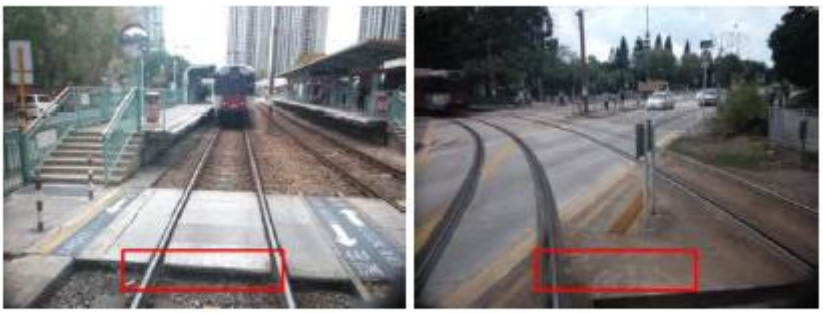
Reference frames with d_r larger than *AVG* $d_{\text{ref frames}}$ are selected and sorted based on their d_r values. In our setting, key frames should not be too close to each other, i.e., the interval of key frames should not be small. Hence, potential key frames are identified based on a key frame interval, $L_{\text{key frame}}$, which is in terms of accumulated average pixel difference and is calculated as,

$$L_{\text{key frame}} = L_{\text{expected}} \times f \times AVG\ d_{\text{ref frames}} \tag{3.5}$$

where L_{expected} is the expected key frame interval in terms of second, f is the frame rate of the studying reference sequence, which is usually set to 25 fps, and *AVG* $d_{\text{ref frames}}$ is computed using Eq. 3.4 without the round up operation. Note that "average pixel differences" is used to classify the frame motion as mentioned in section 3.1.1.1. We also infer the travel distance through accumulating the "average pixel differences". Therefore, we can have certain control of the closeness of the key frames.

Reference frames with larger d_r mean that there are larger variations in the texture in the red-bounded region as shown in Figure 3.4. It usually happens when the vehicle is crossing a vehicle-pedestrian interface (Figure 3.6a) or turning left/right (Figure 3.6b). This kind of frames usually have obviously different structures or skeleton from other frames in the same

reference sequence and there are also more representative features such as road signs. Hence, these frames are suitable for selecting as key frames.



(a) (b)

Figure 3. 6 (a) Reference frame with the largest d_r = 111.73, Frame 251 (b) Reference frame with the second largest d_r = 103.54, Frame 44

3.1.1.4 Potential Key Frame Evaluation

As the key frames are used to confidently tell whether we arrive at a specific location, they should be obviously different from the other reference frames. Hence, each potential key frame is compared with all other frames using low-resolution whole frame descriptor matching with a search window. A weak potential key frame would have similar structural features with other potential key frames, which would also be reflected by the matching results. The search window is established for considering reasonable viewpoint invariance. The original size of a frame is 640 × 480 and we preserve 5 pixels from each edge for establishing the search window. A frame is downsampled to 64 × 48 and divided into normalized regions with size of 8 × 8. The operation of low-resolution whole frame descriptor matching is similar to Eqs. 3.1-3.3. Let $d_{p,i}$ denote the average pixel difference between each potential key frame and all other reference frames. Then, we compute the average and variance of the average pixel differences of each potential key frame, *AVG* d_p and *VAR* d_p respectively.

$$AVG\ d_p = \frac{\sum_{i=0}^{F-1} d_{p,i}}{F}, p \in [0, P-1] \tag{3.6}$$

$$VAR\ d_p = \frac{\sum_{i=0}^{F-1}(d_{p,i} - AVG\ d_p)^2}{F-1}, p \in [0, P-1] \quad (3.7)$$

where F and P are the total number of frames and potential key frames in the studying reference sequence respectively.

Based on Eqs. 3.6, 3.7, θ_p of each potential key frame is calculated:

$$\theta_p = \frac{VAR\ d_p}{AVG\ d_p}, p \in [0, P-1] \quad (3.8)$$

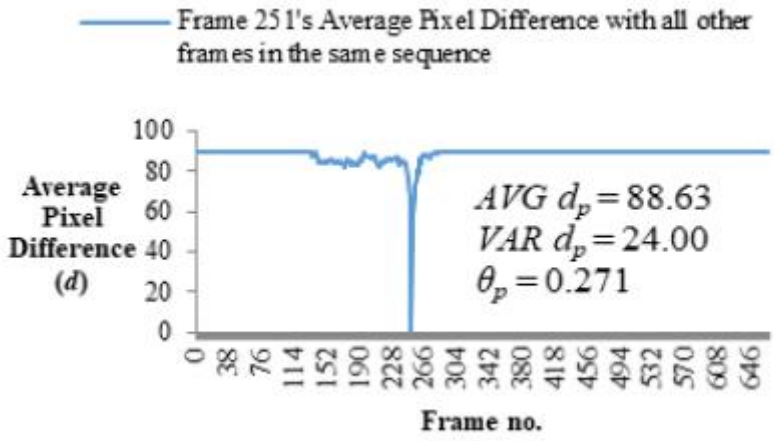


Figure 3. 7 Frame 251's average pixel difference with all other frames (θ_p=0.271, the maximum value of d_p is set to 90 for better visualization)

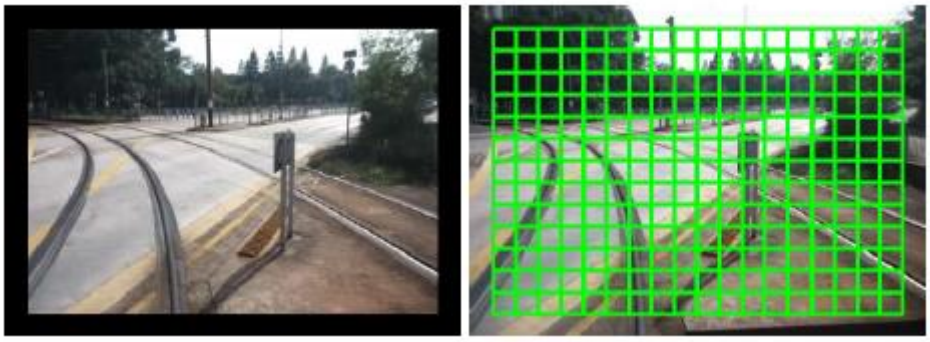
Figure 3.7 shows a persuasive key frame with large *AVG* d_p and small *VAR* d_p. Large *AVG* d_p means that the potential key frame is different from other potential key frames. Small *VAR* d_p also makes sure that the potential key frame would not be similar to other potential key frames located at distant locations. K-means clustering [44] is applied to θ_p to split the potential key frames into 2 groups. One group is with small θ_p while the other is with large θ_p. The potential keys with small θ_p are then identified as official key frames and the list of extracted key frames is finalized.

3.1.1.5 Analysis of Key Frame

Input data:	Y component of a frame
Frame size:	640 × 480 pixels
Cell size:	8 × 8 pixels
Bin size:	9-bin histograms (unsigned gradient)
Block size:	32 × 32 pixels (4 × 4 cells)
Feature vector length:	Each block has a 144-length vector

Table 3. 1 Related Parameters for HOG FV formation

We make analysis of each key frame by using Histogram of Oriented Gradients feature vectors (HOG FV) with the standard formation mentioned in [45]. Table 3.1 lists out the parameter setting for the HOG FV formation.



(a) (b)

Figure 3. 8 (a) Sample image with RoI (Region of Interest) (b) RoI – Green blocks (234 blocks in the RoI)

Figure 3.8a shows a sample image with the defined Region of Interest (RoI). We observe that features on the edges may be difficult to be observed in both reference and testing sequences due to the capture intervals of the frontal camera and the location of the camera. Therefore, we define the RoI in which only the green blocks enclosed by the grids are being processed as shown in Figure 3.8b. Note that there are 234 blocks in our RoI with the pre-defined parameter setting.

3.1.1.5.1 Local Effectiveness

(a) (b)

Figure 3. 9 (a) Three feature blocks in the RoI which represent three different kinds of features, namely sky (blue), trees (green), and road sign (red) (b) same place as (a) but at nighttime and blur situation

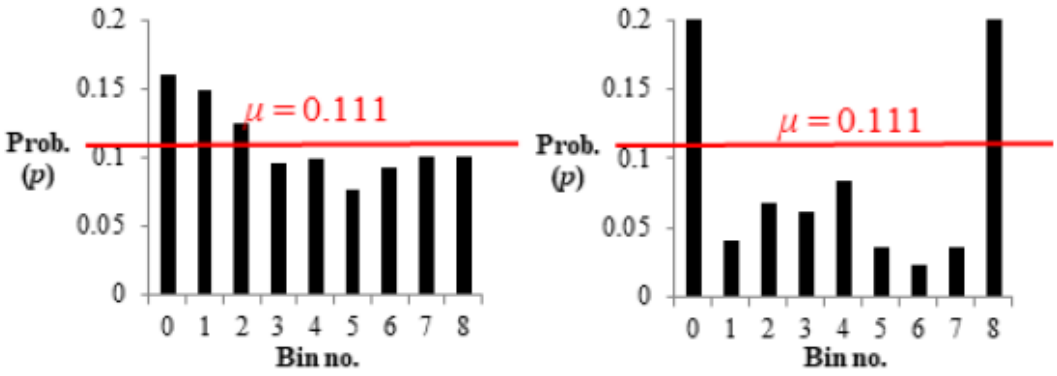
We define features in a frame with high "local effectiveness" if they are with rich texture and strong alignment. In contrast, features with plain texture and irregular alignment are classified as ineffective local features. Figure 3.9a shows three feature blocks in our RoI which represent three kinds of features, i.e., sky (blue block), trees (green block) and road sign (red block). Note that the block size is 32 × 32 pixels, and there are 32 × 32=1,024 values of gradient magnitude and orientation of each feature block respectively.

To show the "local effectiveness" of the feature blocks, we can examine their values of gradient magnitude and orientation. The values of the average gradient magnitude (M) of the three feature blocks are 1.21 (sky), 118.14 (trees) and 81.57 (road sign) respectively. It is obvious that the sky block has plain texture and no edge information, so small value of M is computed. For the trees and road sign blocks, we can see that their values of M are large which means the blocks containing rich texture and edge information. Note that each feature block has one value of M and we have 234 feature blocks for every frame in our RoI, please refer to Figure 3.8b for the RoI. Also, it is clear that trees and road signs include irregular and regular feature patterns respectively. From Figure 3.9b, the trees block (green) cannot be observed at nighttime, hence this kind of features should not be used as effective features to represent a

frame. Figure 3.10a and b show the probability distributions of gradient orientation of the trees feature block (green) and road sign feature block (red) in Figure 3.9a. Let $p_{bi,b}$ be the calling probability of bin *bi* of block *b* (for *b*=0, …, 233). For block *b*, its probability distribution of the gradient orientation can be evaluated in terms of the variance of the probability distribution, $Var(D_b)$ which is computed as:

$$Var(D_b) = \frac{\sum_{bi=0}^{B-1}(p_{bi,b}-\mu)^2}{B-1} \tag{3.9}$$

where *B* is the bin size and μ is the average calling probability which equals to 1/*B*. The $Var(D_b)$ values of the trees and road sign blocks are 0.00077 and 0.00153 respectively. From Figure 3.10a and b, the trees block (a) contains irregular patterns and have small $Var(D_b)$ as all the bins have similar calling probabilities. On the other hand, the road sign block (b) often has higher probabilities on particular bins because of its regular pattern.



(a) (b)

Figure 3. 10 (a) Probability distribution of gradient orientation of trees feature block (green) in Figure 3.9a (b) Probability distribution of gradient orientation of road sign feature block (red) in Figure 3.9a

We combine the average gradient magnitude (*M*) and the variance of probability distribution $Var(D_b)$ to compute the local effectiveness of each feature block, $E_{L,b}$, as follows.

$$E_{L,b} = M_b \times Var(D_b) \times w_b \tag{3.10}$$

where w_b is a weight for block b which reflects the distance between the camera and block b. From Figure 3.8b, it is not difficult to infer those blocks located at the lower part are closer to the camera and this region is less likely to be blocked by dynamic objects during recognition. Therefore, high weight is set for blocks located at the lower part, in which features closer to the camera can be observed clearly and with less blur by because of the highlights of the vehicle in the evening, etc. The $E_{L,b}$ values are normalized by dividing by the highest value of the local effectiveness in the current studying frame.

3.1.1.5.2 Global Effectiveness

Features with high local effectiveness, $E_{L,b}$, are not necessarily to be effective and representative with respect to the entire reference sequence, i.e., all other frames. Some features with high $E_{L,b}$ could be repeated in other frames in the same reference sequence, hence these features are not representative. We propose "Global Effectiveness" to evaluate the features by comparing to all other frames in the same reference sequence. We define that the unique features of a frame should be the features with both high local and global effectiveness. Features with high global effectiveness mean that these features can only be observed in a specific frame but not the other frames in the same reference sequence. Cosine distance measure (C) and Block-to-Block HOG feature vector comparison are used for obtaining the global effectiveness.

$$C = 1 - \cos\omega = 1 - \frac{\mathbf{p}\cdot\mathbf{q}}{\|\mathbf{p}\|\|\mathbf{q}\|} \tag{3.11}$$

where $\cos\omega$ is the Cosine similarity which is defined as the cosine of the angle difference between vectors $\mathbf{p}$ and $\mathbf{q}$.

To obtain the global effectiveness, all feature blocks (234 blocks per frame in the RoI) of a key frame are compared to the corresponding block locations in all other frames. There

could be undesirable or slight shifts of features in practice, hence we look for $C_{\min}$ which is the minimum Cosine distance within a search window of the current studying feature block.

$$C_b = C_{min} = \underset{k\in[0,K-1]}{\arg\min}\, C_k \tag{3.12}$$

where C_k is the Cosine distance of the k^{th} search location within the search window and C_b is the final Cosine distance of a comparison of block b. To find C_b, please refer to Eqs. 3.1-3.3. The only difference is that we use Cosine distance as the cost function for the comparison instead of average pixel difference. We set the search range to ±2 cells and consider 1 cell shift for each search location. There are $(2 \times 2 + 1)^2 = 25$ search locations in a search window and, this pre-defined search range is sufficient to cover the consideration of the feature shifts in our experiments. The average and variance of the Cosine distances of each feature block, $AVG(\mathbf{C}_\text{b})$ and $VAR(\mathbf{C}_\text{b})$ can be calculated as:

$$AVG(\mathbf{C}_\text{b}) = \frac{\sum_{i=0}^{F-1} C_{b,i}}{F} \tag{3.13}$$

$$VAR(\mathbf{C}_\text{b}) = \frac{\sum_{i=0}^{F-1} (C_{b,i} - AVG(\mathbf{C}_\text{b}))^2}{F-1} \tag{3.14}$$

$$\tau_{C,b} = \frac{VAR(\mathbf{C}_\text{b})}{AVG(\mathbf{C}_\text{b})} \tag{3.15}$$

where $C_{b,i}$ is the Cosine distance of block b compared to frame i, F is the total number of frames in the studying reference sequence and $\tau_{C,b}$ is the ratio of $VAR(\mathbf{C}_\text{b})$ to $AVG(\mathbf{C}_\text{b})$ of block b. Note that a standard global effective feature block should have large $AVG(\mathbf{C}_\text{b})$ and small $VAR(\mathbf{C}_\text{b})$, as the feature block should have large distance from the blocks in all other frames and all its distances should keep at a large value level. The values of $\tau_{C,b}$ are then normalized and converted into $E_{G,b}$ as follows.

$$E_{G,b} = \frac{\min \tau_C}{\tau_{C,b}} \tag{3.16}$$

where min τ_C is the smallest $\tau_{C,b}$ in the current studying frame.

Figure 3.8a shows a sample image with the defined Region of Interest (RoI). We observe that features on the edges may be difficult to be observed in both reference and testing sequences due to the capture intervals of the frontal camera and the location of the camera. Therefore, we define the RoI in which only the green blocks enclosed by the grids are being processed as shown in Figure 3.8b. Note that there are 234 blocks in our RoI with the pre-defined parameter setting.

3.1.1.5.3 Mixed Effectiveness

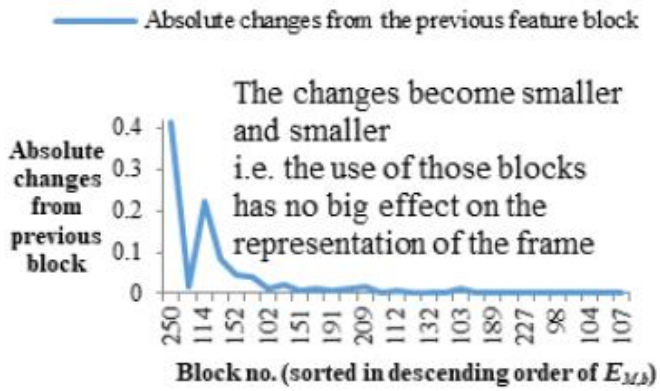
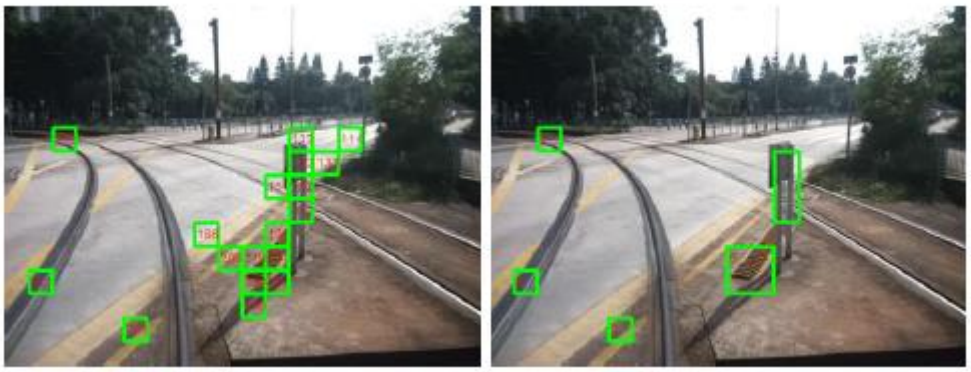


Figure 3. 11 Absolute changes from the previous feature block after sorting based on $E_{M,b}$

As abovementioned, a real effective and representative feature block should have both high local and global effectiveness. We introduce the mixed effectiveness ($E_{M,b}$) as the product of $E_{L,b}$ and $E_{G,b}$. We normalize $E_{M,b}$ by means of dividing by the highest value of $E_{M,b}$ in the current studying frame. Feature blocks are sorted based on $E_{M,b}$ and Figure 3.11 shows the absolute changes from the previous feature block of frame i = 44 in sequence – LRTS1. The absolute change is the drop in the mixed effectiveness of the sorted feature blocks. The first feature block should be the beginning most effective blocks to represent the frame. It is obvious that the absolute changes for the end of Figure 3.11 become smaller and smaller. This implies that the use of these ineffective blocks has no positive effect on the representation of the frame in terms of the value of effectiveness. We set a threshold $T_{\text{cut off}}$ = 0.001 for detecting the cut

off point of the ineffective blocks. The cut off point is detected when the absolute change between two consecutive feature blocks is smaller than $T_{\text{cut off}}$. The final list of effective feature blocks is then finalized. Figure 3.12a displays the extracted effective feature blocks of frame i = 44 based on $E_{M,b}$ and the absolute changes in $E_{M,b}$. By using the proposed mixed effectiveness, only few but the most effective feature blocks are required to represent a frame with respect to the entire reference sequence. The most effective and representative features in frame i =44 shown in Figure 3.12a is clearly the road sign located at a around the middle of the frame. It is important to note that we do not intentionally extract the artificial features like traffic signs and lane as the effective features.



(a) (b)

Figure 3. 12 (a) Final extracted effective feature blocks of frame i = 44 based on $E_{M,b}$ and the absolute changes in $E_{M,b}$ (b) Final extracted effective feature patches of frame i = 44

3.1.1.5.4 Effective Feature Blocks Grouping

Refer to Figure 3.12a, we can see that most of the effective feature blocks are connected, hence we combine them to form different effective feature patches. Note that individual effective feature block can be regarded as small feature patterns for feature matching, but it could be easily to have false positive for small feature patterns. On the other hand, larger feature patterns, i.e., feature patches, are more expressive and larger patters reduce the probability of getting false positives. Therefore, each feature patch contains one to few feature blocks, please refer to Figure 3.12b for an example of grouping effective feature blocks. Each effective feature

patch is shifted to form a number of shifted versions. This is our proposed offline feature-shifts approach for feature matching during online practicing stage. Different shifted versions of the feature patches are then stored in the database for later usage. We set the shift range to ±3 cells (3 × 8 = 24 pixels), which means we shift a patch horizontally and/or vertically by 4 pixels for each version. Hence, there are in total $(2 \times (3 \times 8/4) + 1)^2 = 169$ shifted versions and each version generates a HOG feature vector [45].

3.1.2 Online Scene Recognition and Tracking Stage

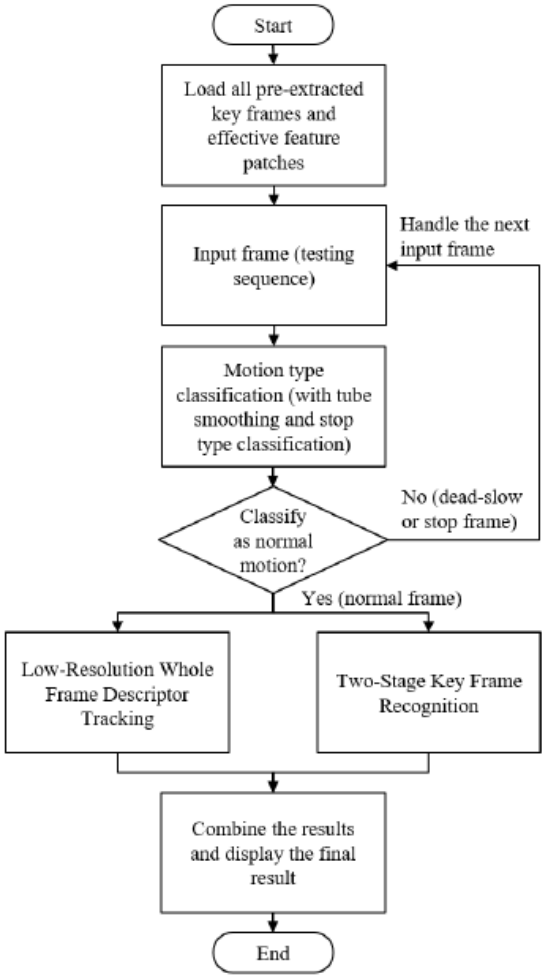


Figure 3. 13 Flowchart of online scene recognition and tracking stage

Figure 3.13 shows the workflow of our proposed online scene recognition and tracking stage (i.e., online practicing stage). The first step of the online practicing stage is to load all the

learned key frames and reference frames from the abovementioned offline shallow learning stage into the system. We propose a two-stage strategy to simplify and speed up the key frame recognition. We boost the recognition performance by introducing a search window weighting approach. We also suggest a reference frame tracking, which contributes to the efficiency of the recognition system. By considering both the estimated and computed tracking results, the temporal relationship between neighbouring frames is utilized, in which the recognition consistency can be enhanced. The details of the key frame recognition and reference frame tracking are discussed as follows.

3.1.2.1 Key Frame Recognition

3.1.2.1.1 Offline Feature-Shifts Approach

As mentioned in section 3.1.1.5.4, we consider the practical feature-shifts of the key frame patches in an offline manner for the sake of efficiency. We build a shift window for each key frame patch and shift the patch inside the shift window to form different versions of the feature patch. During testing, we compute the feature vector of each patch in the current query frame based on the initial locations of the patches of the current studying key frame. The feature vector of each patch in the current incoming query frame is compared with all versions of the feature vectors of the corresponding patch stored in the database by using Cosine similarity (Eq. 3.11). For every single patch, we look for the best matched location ($\Delta u_{k,\min}$, $\Delta v_{k,\min}$) among all the shifted versions which offers the highest similarity score as follows.

$$(\Delta u_{k,\min}, \Delta v_{k,\min}) = \underset{\Delta u_k, \Delta v_k \in [-\frac{sh}{st}, \frac{sh}{st}]}{\arg\min} C(FV_{k,\text{query}}, FV_{k,\text{key},\Delta u_k,\Delta v_k}) \tag{3.17}$$

where $FV_{k,\text{query}}$ is the feature vector of the k^{th} patch in the current incoming query frame and $\text{FV}_{k,\text{key},\Delta uk,,\Delta vk}$ is the feature vector of the shifted version of the k^{th} effective feature patch in the current comparing key frame with coordinates (Δu_k, Δv_k). *sh* and *st* are the shift range and

stride in terms of pixels. The final Cosine similarity between the studying patches, C_k, is calculated as:

$$C_k = C(FV_{k,\text{query}}, FV_{k,\text{key},\Delta u_{k,\min},\Delta v_{k,\min}}) \tag{3.18}$$

3.1.2.1.2 Shift Window Weighting Approach

It is difficult to detect the global peak, i.e., whether we recognize the key frames, in real time as we could be trapped into the local peaks which are located before the global peak. To avoid this kind of situations, the similarity (or Weighted Confidence Level to be defined later) of the global peak and the others should be as different as possible so that we have a clearer decision boundary for decision making. We define the tolerance of the key frame recognition of a key frame as the difference between the global peak similarity and the average similarity of the matching curve. Large tolerance means that we have a sharper decision boundary which is important to recognition and navigation. We have carefully studied the key frame recognition results of each key frame. We are targeting at the original effective feature patches we found at the offline learning stage previously, even that we have already shifted the patches for practical feature-shifts in real time. From the actual matching results, we realize that patches are matched to different shifted versions, and we study the distributions of the matched locations of the feature patches.

Figure 3.14 shows an example of a shift window with u and v range from -4 to 4. $d_{\max}$ is the largest L2 norm distance [45] from the origin in the shift window, sqrt(4^2 + 4^2) = sqrt(32) in this case. Based on $d_{\max}$, the distance between each shift point and the origin can be normalized and the weighting for the k^{th} patch, w_k, can be calculated,

$$w_k = 1 - \frac{d_{(\Delta u_{k,\min},\Delta v_{k,\min})}}{d_{\max}} \tag{3.19}$$

where the coordinates ($\Delta u_{k,\min}$, $\Delta v_{k,\min}$) are computed using Eq. 3.17 and we can calculate the distance between ($\Delta u_{k,\min}$, $\Delta v_{k,\min}$) and the origin of the k^{th} effective feature patch. If the distance is small, the patch is matched to the original effective feature patch and we aware this matching by assigning a larger weight. The Weighted Confidence Level (*WCL*) between two frames is defined as the weighted sum of the similarities between all respective patches.

$$WCL = \frac{\sum_{k=0}^{K-1} C_k w_k}{K} \tag{3.20}$$

where K is the number of effective feature patches in a key frame, C_k and w_k are obtained using Eqs. 3.18 and 3.19 respectively.

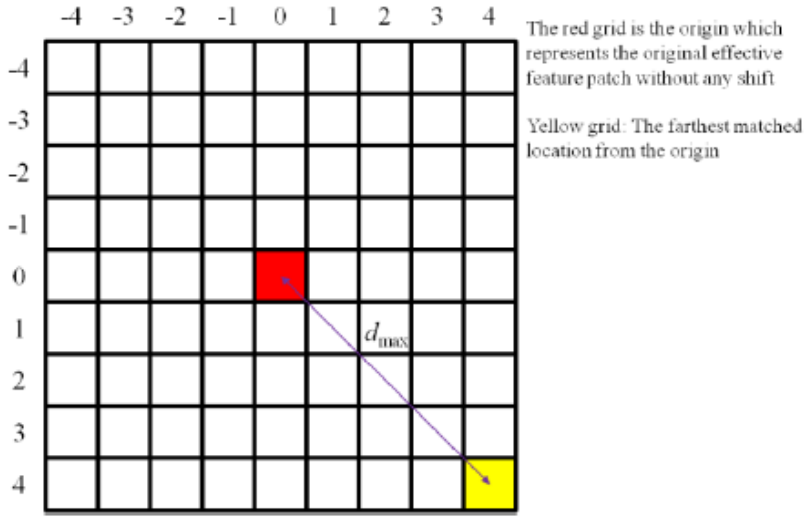


Figure 3. 14 An example of shift window with u and v range from -4 to 4. The red grid is the origin which represents the original effective feature patch without nay shift. The yellow grid is the farthest matched location from the origin

3.1.2.1.3 Two-Stage Key Frame Recognition

In the proposed offline key frame identification, we use low-resolution whole frame descriptor to select potential key frames by means of comparing the structural features of all the frames in the studying reference sequence. The same concept is used to recognize key frames. The comparison between frames is based on the average pixel differences, refer to Eq. 3.1, which is then converted into similarity score, *SS*, as below.

$$SS = \begin{cases} 0 & , if\ apd > upper\ bound \\ \frac{upper\ bound - apd}{upper\ bound - lower\ bound} & , otherwise \end{cases} \quad (3.21)$$

where *apd* is the average pixel differences between the current frame and the studying key frame. For unsigned 8-bit image, the largest value of *apd* is 255 but our experimental results show that most of the differences vary between 70 and 120. Hence, the *upper bound* of *apd* and *lower bound* of *apd* are set to 70 and 120 respectively.

We also proposed to split our online key frame recognition into two stages in order to take the advantage of efficient low-resolution whole frame matching. The first stage is low-resolution whole frame descriptor matching, targeting at fast and concise decision. The second stage is the abovementioned patch-based key frame recognition which employs the offline feature-shifts with shift window weighting approach. The operation of the two-stage key frame recognition is described as follows.

$$SS = \begin{cases} \text{the best match is found directly} & , if\ SS > T_{matched} \\ \text{no further operation of key frame recognition } (MCL = 0) & , if\ SS < T_{unmatched} \\ \text{perform patch} - \text{based key frame recognition } (MCL = WCL \times SS) & , otherwise \end{cases} \quad (3.22)$$

where *WCL* (weighted confidence level) and *SS* (similarity score) are computed using Eqs. 3.20 and 3.21 respectively. Our logic is that if the *SS* shows enough confidence in the current match, the best match to the comparing key frame is found directly. If *SS* is very low, the current match is not likely to be the best match. The operation of the key frame recognition for the current query frame would be ended immediately to save the time cost. For *SS* without clear indication of the match, our patch-based key frame recognition would be activated, and the current match would be evaluated by the modified confidence level, *MCL*, for which it is the product of *SS* and *WCL*. To get a high value of *MCL*, both *SS* and *WCL* should be high. This implies that both structural features and local feature patches are employed to ensure a reliable level of confidence. Note that not all the matches have the corresponding *MCL*, so the

default value of *MCL* is 0. We assume that the reference key frames appear in order as the starting point of the vehicle is known. A parallel key frame recognition scheme is also adopted to avoid trapping into false negatives. This means that the current key frame and the next key frame are under the matching process concurrently. When the current key frame is accidentally matched with lower confidence because of practical reasons such as dynamic objects and blurring, the next key frame is useful to rectify the current position of the vehicle when it is matched. The core idea of key frame recognition is that we match some critical landmark locations, i.e., key frames, to lock the current position of the vehicle and then the reference frame tracking can be performed to trace the recent positions of the vehicle until the next landmark location is recognized.

3.1.2.2 Reference Frame Tracking

3.1.2.2.1 Low-Resolution Whole Frame Descriptor Tracking

During the online practicing stage, if the input query frame is classified as dead-slow or stop frame, please refer to Eqs. 3.1-3.3 and Figure 3.13, no operation on tracking and recognition would be performed to save the cost and the previous result is kept. For an incoming input frame which is classified as frames with normal motion, low-resolution whole frame descriptor tracking with our proposed tube of frames concept is performed to find the best match to one of the reference frames within a search range (ζ) and use the previous results to predict the next best match. The down-sampled input frame is compared to a set of reference frames using average pixel difference, refer to Eq. 3.1. In our proposed reference frame tracking, we introduce a novel idea on combining the estimation of the next best match and the evaluation of the current match.

3.1.2.2.1.1 Estimation of the Next Best Match

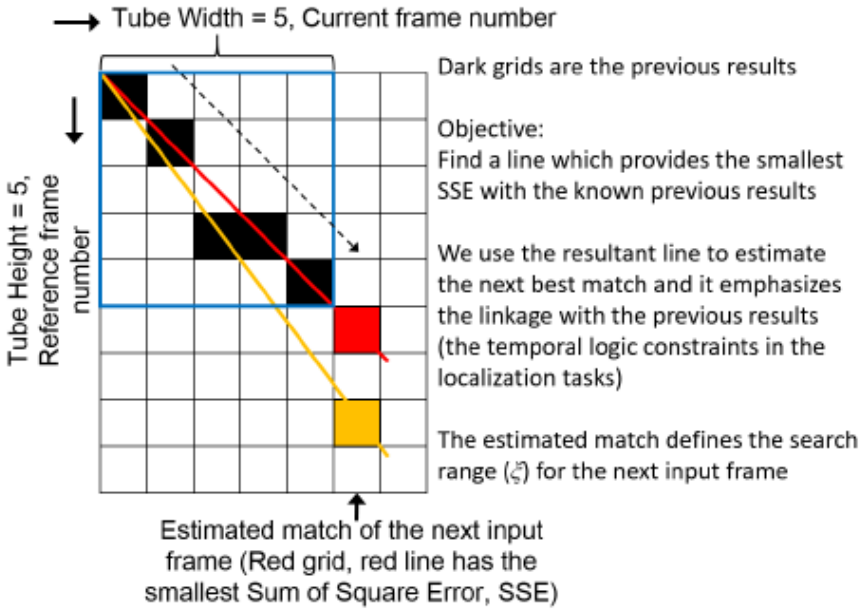


Figure 3. 15 Graphical illustration of a sample case in tube of frames concept and estimation of the next best match

Figure 3.15 shows a graphical illustration of our proposed ideas of tube of frames and the estimation of the next best match. The horizontal axis represents the current incoming query frame number, and the vertical axis represents the reference frame number. Assume that the dark grids are the previous results in which these input and reference frame pairs give the smallest average pixel difference among the corresponding set of reference frames. In this case, the tube width is 5 and the current query frame is the 5^{th} input frame, last column in the blue region. We use the previous 5 results, the 5 dark grids, to predict which pair in the next column, i.e., the next input frame, provides the smallest average pixel difference. We would like to find a line which produces the smallest Sum of Square Error (SSE) with the previous results. We draw a number of lines based on the pre-defined search range and search step, φ_{range} and φ_{step}, which define the slope of each line. The estimated match defines the search range (ζ) for the next input query frame. The tube width is fixed, and the tube keeps sliding based on the final result of each input query frame which is obtained from the estimated match (discuss in the sub-section) and the current calculated match (discuss in the next sub-section).

3.1.2.2.1.2. Evaluation of the Current Match

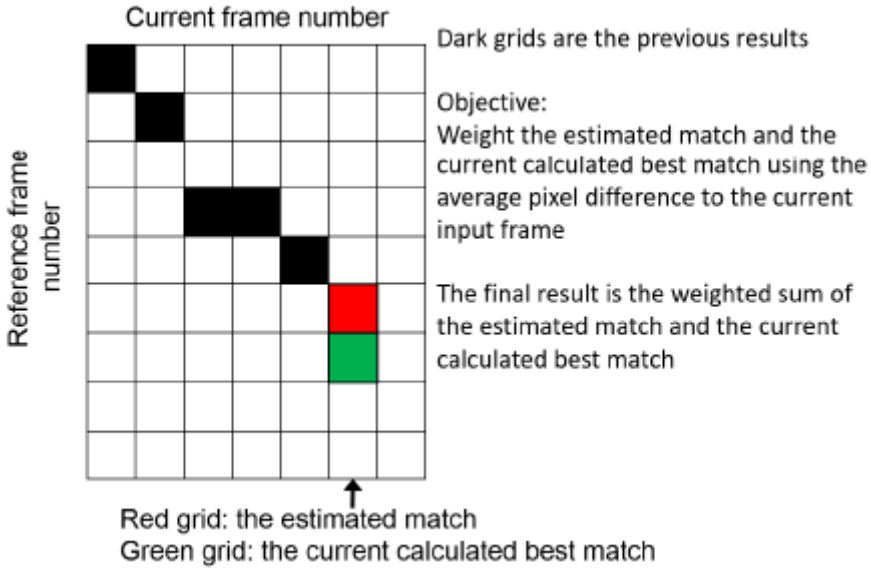


Figure 3. 16 Graphical illustration of evaluation of the current match

We discuss the confidence measure of the estimated and calculated matches. Figure 3.16 shows how to obtain the final result of the current input frame based on the estimated match and the current calculated best match graphically. Suppose that we are at the 6th column and the red grid is the estimated match obtained in the previous input frame. The green grid is the current calculated best match which gives the smallest average pixel differences within the corresponding search range (ζ). The average pixel differences of the estimated match and the calculated match are indicated as $apd_{estimated}$ and $apd_{calculated}$. The two matches are weighted based on their *apd*s to get the final result as below.

$$I_{final} = W_{estimated} I_{estimated} + W_{calculated} I_{calculated} \tag{3.23}$$

where I_{final} is the final result of the current input frame in terms of the reference frame index, $I_{estimated}$ is the estimated match and $I_{calculated}$ is the current calculated match. $W_{estimated}$ and $W_{calculated}$ are the weights of the estimated and calculated matches respectively. The weights are computed as follows.

$$W_{estimated}=\frac{apd_{calculated}}{apd_{estimated}+apd_{calculated}}$$
$$W_{calculated}=\frac{apd_{estimated}}{apd_{estimated}+apd_{calculated}} \quad (3.24)$$

where the confidence of the estimation and the current calculation are based on the average pixel difference. If the current input frame is closer to the estimated match, i.e., lower *apd*, a higher weight is assigned to the estimated match and vice versa.

3.1.3 Experimental Results

3.1.3.1 Datasets

Extensive experiments have been conducted. There are four datasets, and we compare our proposed method to several approaches, namely SeqSLAM [7], [42], ABLE-M [31], [32], and AlexNet [35], [46] conv3 layer feature-based approach. SeqSLAM is the first sequence-based approach which assumes constant speed situation. It acts as a baseline to evaluate the performance of different methods. ABLE-M is also a sequence-based method for which the authors used binary sequence codes to represent sequence of images. AlexNet conv3 layer feature-based method uses the off-the-shelf network as a feature extractor to take the advantages of well generalized deep features. The datasets are discussed in the following.

3.1.3.1.1 Light Rail Transit (LRT) Datasets

Figure 3. 17 Examples from LRT datasets, including changes in lighting conditions, blurring, dynamic objects, and varying speeds

The first two datasets are obtained directly from a public transportation system in Hong Kong, Light Rail Transit (LRT). These are real-life sequences, and each dataset contains 4

sequences of the same route, 3 in the daytime and 1 at nighttime. One of the daytime sequences was used as a reference sequence for the offline learning stage. The datasets involve many practical issues like extreme lighting environments, blurring, varying speeds, and dynamic objects. The first shorter dataset has 623 frames in average, short sequence, LRTS, and the second dataset has 2,566 frames in average, longer sequence, LRTL. Figure 3.17 displays two examples of the LRT datasets. The ground truth, i.e., the match pairs, of these datasets are manually marked based on human inspection.

3.1.3.1.2 Nordland Dataset

Figure 3. 18 Nordland dataset consists of extreme changes in appearance and seasons. Note that two images represent the same place in this example

The third dataset is introduced in [42], named as Nordland dataset, which consists of 4 long railway sequences for four seasons. This dataset focuses on the extreme seasonal changes as shown in Figure 3.18. It assumes that the train travels at constant speed in the four journeys. Hence, the dataset is time-synchronized such that any frame in one of the sequences corresponds to the same frame in the other sequences. Note that our work focuses on general situations with varying speeds rather than extreme changes in appearance, we tested our proposed method on this dataset to highlight the difference between SeqSLAM and ours (FMPR). We extracted 5,950 frames from each sequence in this dataset as long sequences for testing and the “Spring” sequence was used as the reference sequence for the offline learning stage. Also, this dataset contains the tunnel parts which have no features to be matched,

examples can be seen in Figure 3.19. Clearly, low average pixel value of frames can be obtained when the train is inside the tunnel parts. Therefore, we use the average pixel value to detect the tunnel region frames which are regarded as dead-slow or stop frame in our system.

Figure 3. 19 Tunnel parts without any feature, Frames 1900 and 2022, in Nordland dataset

3.1.3.1.3 UA Dataset

Figure 3. 20 UA dataset is non-railway case with changing lighting environments, small changes in viewpoints, and image distortions

The last dataset is a short sequence which was captured on the campus of University of Alberta, Edmonton, Canada [43]. This dataset contains non-railway sequences, and it focuses on the day-night lighting conditions as shown in Figure 3.20. We extracted two sequences from this dataset, one daytime and nighttime sequences. Note that this dataset has also been time-synchronized and the two sequences both have 646 frames. There are also some changes in viewpoints and image distortions. The daytime sequence was used a reference sequence for

offline learning stage. The use of this dataset demonstrates the possible extension of our proposed method to non-railway cases.

3.1.3.2 Evaluation of Different Tube Sizes for FMPR

As mentioned in Section 3.1.2.2, our proposed Fast Monocular Place Recognition (FMPR) approach uses the concept of tube of frames which utilizes the temporal information about the vehicle. Different tube sizes, i.e., different number of frames, from 4 to 40, are evaluated on the four datasets to study the relationship between the tube size and the recognition accuracy. Note that this experiment focuses on the tube size, hence the proposed key frame recognition and the evaluation of the current calculated match are not included. A match pair is regarded as correct if its deviation from the ground truth is fewer than 15 frames and the results are listed in Table 3.2.

Testing sequence	Accuracy with different tube sizes						
	4	6	8	10	20	30	40
LRTS2	0.53	*0.64*	0.54	0.59	0.59	0.49	**0.82**
LRTS3	**0.90**	*0.84*	0.61	0.49	0.82	0.61	0.47
LRTS4	**0.78**	**0.78**	0.72	*0.74*	0.68	0.68	0.72
LRTL2	**0.76**	*0.74*	0.60	0.71	0.06	0.59	0.50
LRTL3	*0.34*	**0.49**	0.02	0.08	0.04	0.18	0.04
LRTL4	**0.81**	*0.67*	0.53	0.02	0.20	0.03	0.40
SUMMER	0.27	*0.76*	**0.82**	0.27	0.30	0.72	0.70
FALL	0.82	*0.83*	*0.83*	0.32	0.31	**0.85**	**0.85**
WINTER	0.07	0.07	0.07	0.07	0.16	*0.26*	**0.28**
UA2	0.43	0.88	0.53	0.91	**1.00**	*0.96*	**1.00**
Average	0.57	**0.67**	0.53	0.42	0.42	0.54	*0.58*

Table 3. 2 Recognition results on different datasets with different tube sizes (for FMPR, the best scores are in **bold** typeface and the second-best scores are in *italics* typeface)

From Table 3.2, we can see that larger tube sizes generally benefit constant speed situations, like UA and Nordland datasets, the time-synchronized datasets. When the tube size is 40, sequences FALL and UA2 achieve 0.85 and 1.00 accuracies respectively. On the other hand, smaller tube sizes perform better in varying speed situations than larger tube sizes. For tube size = 4, sequences LRTS3 and LRTS4 attain 0.90 and 0.78 accuracies respectively. As

varying speed situations also contain similar speed situations, larger tube sizes also work properly in few sequences in the LRTS datasets like LRTS2, 0.82 accuracy with tube size = 40. This reflects that different datasets have their respective optimal tube sizes. For the sake of comparing different methods, we select a tube size which offers the best performance on average. Therefore, for the rest of our experiments, we set the tube size to 6.

3.1.3.3 Evaluation of Key Frame Recognition

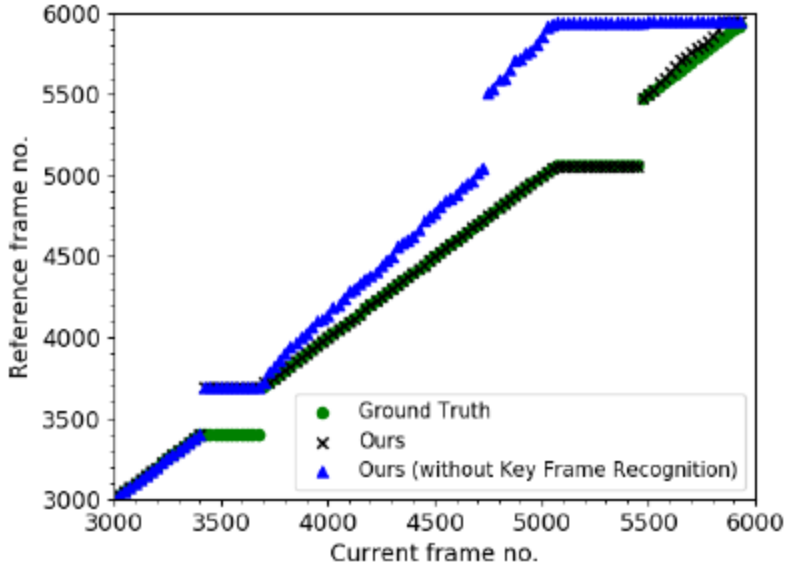


Figure 3. 21 Match pairs from the proposed method with and without key frame recognition for accumulated deviation elimination. For better resolution of the plot, we drew one match pair for each 25 consecutive pairs from Frame 3000 to Frame 5950. (Reference sequence: “Spring”, testing sequence: “Summer”)

The effectiveness of the proposed key frame recognition is important to rectifying the recognition results. As the reference frame tracking is based on both the predicted match pair based on the previous results and the current frame pair matching result, the estimation could deviate from the ground truth and the deviation accumulates over time. The term “deviation” is defined as the difference between the current output and the corresponding ground truth. The main function of the proposed key frame recognition is to eliminate the accumulated deviation. Figure 3.21 shows the matched pairs from the proposed method with and without key frame recognition for accumulated deviation elimination. Each current frame has the corresponding

matched pair from its reference sequence (Ground Truth, green circle). We can see that the shape of the ground truth seems irregular because of the non-uniform vehicle speed situation. Note that more overlaps with the ground truth mean better recognition performance. Without our key frame recognition (blue triangle), obvious deviation from the ground truth can be seen at location where the current frame number = 4000. This means that the blue and green curves are not overlapping starting from the current frame number = 4000. In addition, with the proposed key frame recognition (the black cross), the black and green curves are highly overlapped. This shows the effectiveness of our proposed key frame recognition.

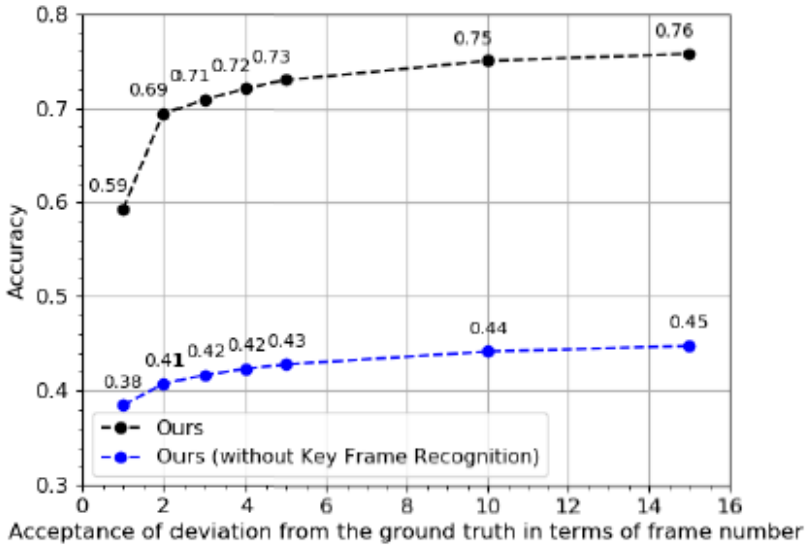


Figure 3. 22 Accuracy of the proposed method with and without key frame recognition under different acceptances of deviation. We have set the range of the vertical axis from 0.3 to 0.8. Grid lines and data labels are also added. (Reference sequence: “Spring”, testing sequence: “Summer”)

Figure 3.22 further shows the accuracies of the proposed method with and without key frame recognition under different acceptances of deviation, the black and blue curves respectively. We also consider a matched pair is correct under different acceptances of deviation from the ground truth for showing the closeness of our recognition results and the ground truth. For example, when the acceptance of deviation is 3 frames, a matched pair is regarded as correct only if the absolute frame difference between the pair and the ground truth

is within 3. It is obvious that the black curve, with key frame recognition, is always above the blue one, without key frame recognition. This shows that the proposed key frame recognition could fulfil its core function in this typical case.

3.1.3.4 Comparison of SeqSLAM

We compared our proposed method with SeqSLAM by using OpenSeqSLAM package [7], [42]. We used the default parameters of the package, except for the reduced image size was changed from 64 × 32 to 64 × 48 pixels. The comparison was done via the precision value *P*, entitled as accuracy, which is the percentage of correctly matched pairs of all reported pairs for matching. Nevertheless, using precision alone may not be too fair since one may make evaluation by neglecting some number of pairs with low confidence, which offers higher precision score substantially. Hence, we use F1 score [47] for the evaluation, which is defined as:

$$F1 = 2 \times \frac{P \cdot R}{P + R} \tag{3.25}$$

where *P* is the precision and *R* is the recall rate which is the percentage of available matched pairs to the total number of pairs for performance evaluation. Different values of *R* can be obtained by using a threshold to filter the pairs with low confidence. High F1 score means better performance, both *P* and *R* have to be high to obtain high F1 score. In this setting, methods offer low recall rate resulting in low F1 score, i.e., just selecting a small number of high confidence pairs for evaluation is reflected by the F1 score. Note that SeqSLAM uses a simple threshold to eliminate weak matches for generating different sets of precision and recall rate, and the default value of the threshold is 0.9. In our proposed method, the recall rate *R* is always 1.0 which means all the pairs are used for the evaluation. Our goal is to maximize the precision at 100% recall rate.

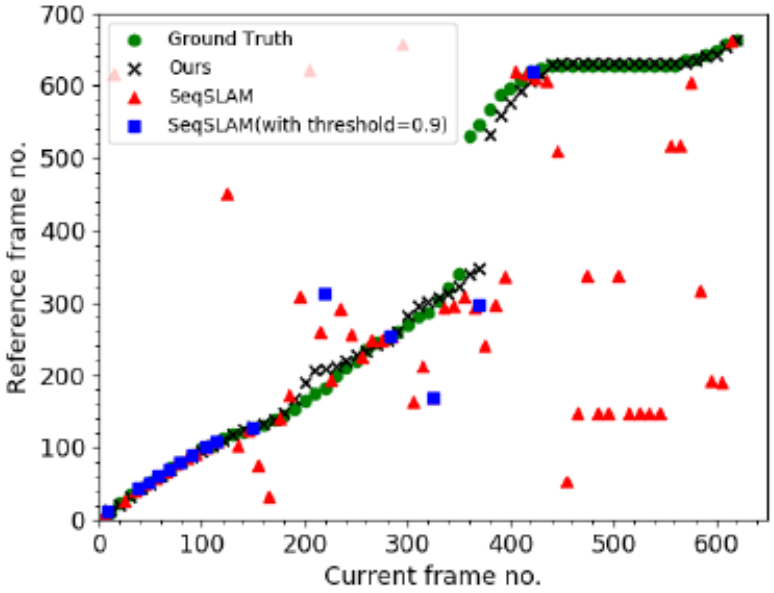


Figure 3. 23 Match pairs from the proposed method and SeqSLAM. For better resolution of the plot, we drew one match pair for each 10 consecutive pairs. (Reference sequence: "LRTS1", testing sequence: "LRTS3")

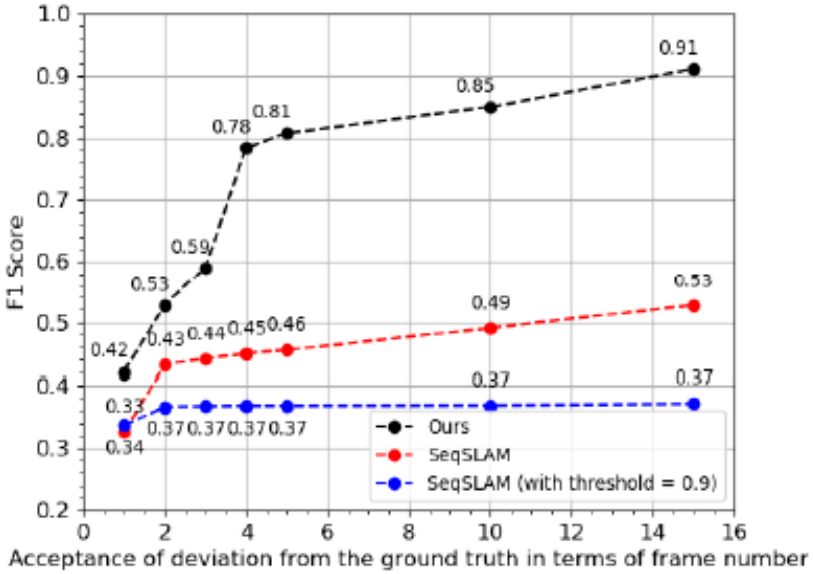


Figure 3. 24 F1 scores of the proposed method and SeqSLAM. (Reference sequence: "LRTS1", testing sequence: "LRTS3")

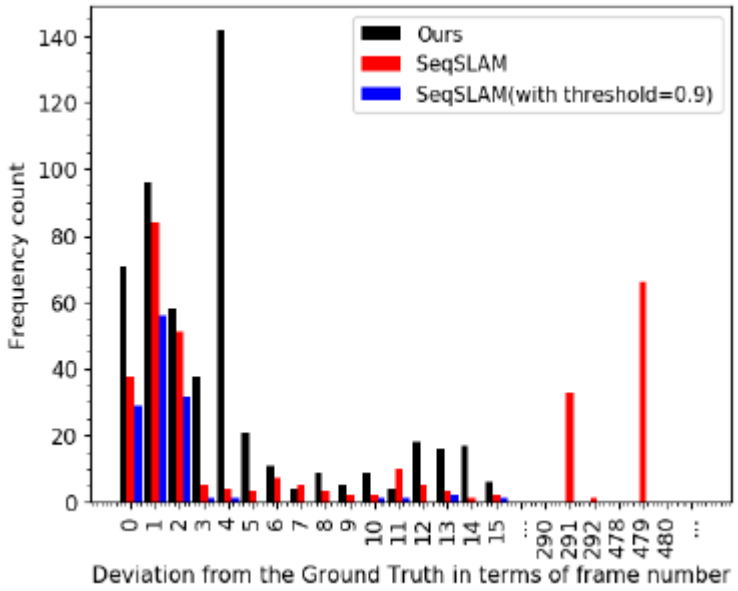


Figure 3. 25 Distribution of deviation of the proposed method and SeqSLAM. (Reference sequence: "LRTS1", testing sequence: "LRTS3")

The black crosses form two separate curves and the red triangles as shown in Figure 3.23 display the matched pairs from the proposed method and SeqSLAM respectively. It is clear that SeqSLAM is not designed to handle varying speed situations which is reflected by the scattered red triangles. On the other hand, our proposed method, the black crosses, uses tube of frames, is more suitable for handling varying speed situations. Figure 3.24 shows the corresponding F1 scores of the results. Consider that we accept 15-frame deviation from the ground truth, our proposed method achieves F1 with 0.91 (black) while SeqSLAM only obtains 0.53 (red) and 0.37 (blue) respectively. The distribution of deviation from the ground truth of ours (FMPR) and SeqSLAM are shown in Figure 3.25. Note that SeqSLAM with thresholding (blue) has only 150, out of 668 available match pairs for evaluation, while both ours (black) and full SeqSLAM (red) attained 100% recall rate. All of the matched pairs of ours (black) have only little deviation from the ground truth. In addition, it is obvious that the full SeqSLAM (red) gives many mismatched pairs, there are two red bars on the right-hand side with high frequency counts. Specifically, there are 33 and 66 pairs deviating from the ground truth with

291 and 479 frames respectively. This is consistent to the scattered red triangles as shown in Figure 3.23.

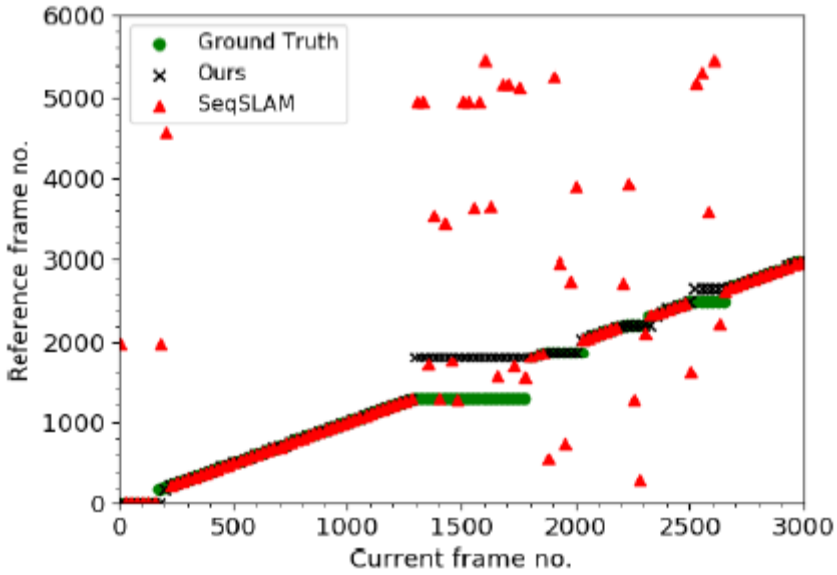


Figure 3. 26 Match pairs from the proposed method and SeqSLAM. We drew one match pair for each 25 consecutive pairs from Frame 0 to Frame 3000. (Reference sequence: "Spring", testing sequence: "Fall")

Figure 3.26 shows another example that SeqSLAM cannot effectively deal with varying speed situations. SeqSLAM cannot report the correct matched pairs at the flat regions. The red triangles do not overlap with the ground truth in particularly the flat regions, around Frame 1500 to 2000. This reflects the problem of constant speed assumption in SeqSLAM. The flat regions are the tunnel parts and stop locations in the Nordland dataset. Please refer to Figure 3.19 for the examples of tunnel parts in this dataset.

3.1.3.5 Comparison of ABLE-M

Apart from SeqSLAM, we also compared our method with ABLE-M, OpenABLE [31], [32]. We used the default setting of OpenABLE and each frame was downsampled to 64 × 64 for forming the Local Difference Binary (LDB) descriptor. We set the sequence length to 20 and the threshold for reporting confident match pairs to 0.35. This means that 20 frames are grouped as a sequence to form the binary sequence codes. If the similarity between two

sequence codes is higher than 0.35, a confident match pair is reported. Note that the main contribution of ABLE-M is about using the binary sequences as features to build the similarity matrix. They simply took the element with the highest similarity at each column as the output of the matched pair and they do not use any temporal logic constraint on searching the similarity matrix.

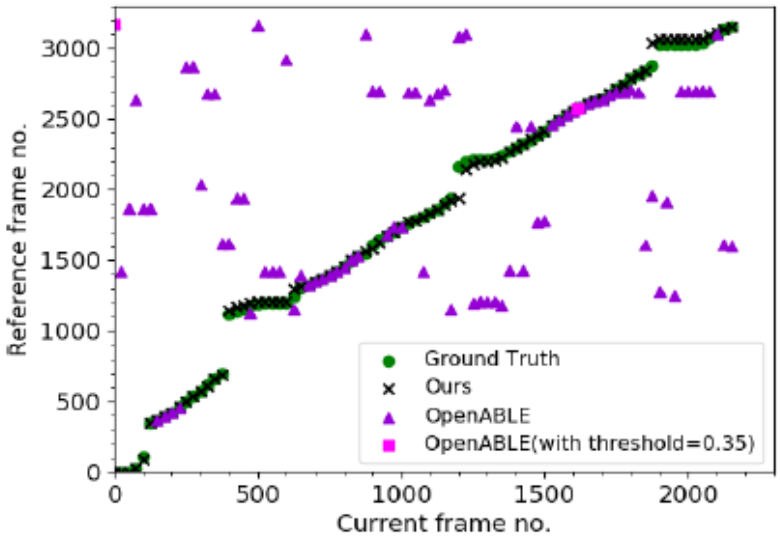


Figure 3. 27 Match pairs from the proposed method and ABLE-M. We drew one match pair for each 25 consecutive pairs from Frame 0 to Frame 2172. (Reference sequence: "LRTL1", testing sequence: "LRTL2")

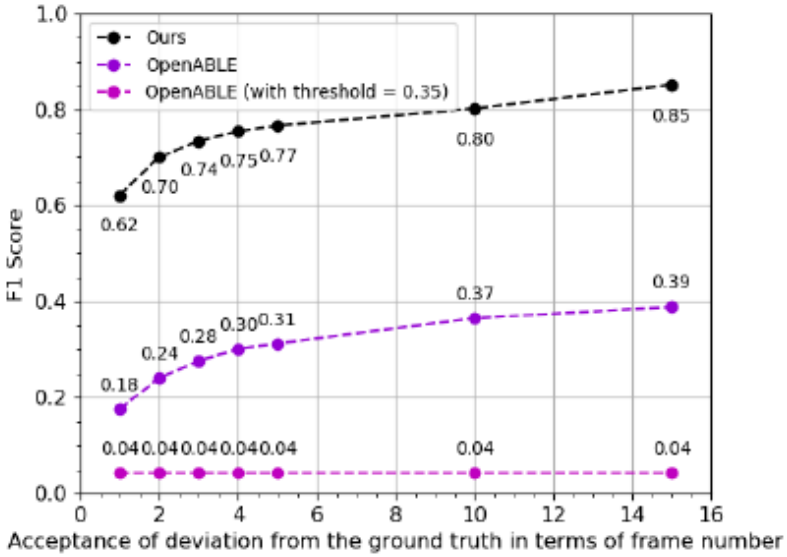


Figure 3. 28 F1 scores of the proposed method and ABLE-M. (Reference sequence: "LRTL1", testing sequence: "LRTL2")

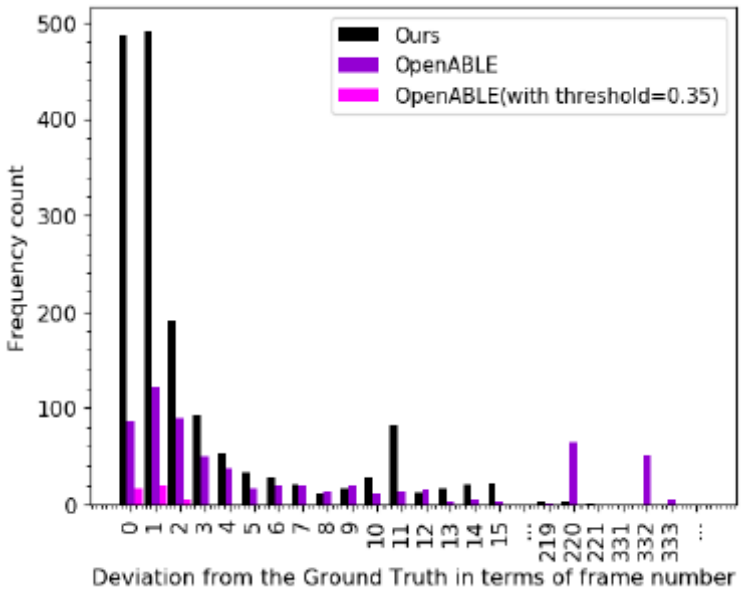


Figure 3. 29 Distribution of deviation of the proposed method and ABLE-M. (Reference sequence: "LRTL1", testing sequence: "LRTL2")

Figure 3.27 shows the matched pairs from the proposed method (black cross) and ABLE-M (purple triangles and pink squares). The testing sequence, LRTL2 is a nighttime sequence with extreme changes in illumination and blurring. ABLE-M cannot effectively handle the changing lighting conditions, please refer to the scattered purple triangle. In addition, ours can deal with this nighttime sequence without any obvious problem. Therefore, ours, the black crosses, is highly overlapped with the green circles, ground truth. Figure 3.28 displays the F1 scores of ABLE-M and ours for this nighttime testing sequence. Note that OpenABLE with threshold = 0.35 suffers from very low recall rate, i.e., 2.21%, 48 out of 2172 frames. For the case of 15-frame deviation from the ground truth, ours attains F1 score = 0.85 with 100% recall rate compared to 0.39 achieved by OpenABLE at 100% recall rate. Figure 3.29 also gives the distribution of deviation from the ground truth of ABLE-M and ours. Similar observation as the previous comparison of SeqSLAM, ABLE-M with the default threshold has the problem of low recall rate. We can also observe that 100 match pairs from ABLE-M, purple bars, have 220-frame and 332-frame deviation from the ground truth.

3.1.3.6 Overall F1 Score and Time Cost Comparisons

Total no. of frames in database	Total no. of frames in testing seq.	Testing sequence	Ours (FMPR)	SeqSLAM [7],[42]	ABLE-M [31],[32]	AlexNet conv3 [13],[35],[46]	NetVLAD [2],[3]	CALC [36]	Ours (without Key Frame Recognition)
669	617	LRTS2	0.782	0.741	0.491	0.907	0.747	0.153	0.782
	634	LRTS3	0.911	0.528	0.917	0.980	0.988	0.729	0.945
	580	LRTS4	0.877	0.592	0.828	0.927	0.920	0.809	0.851
3169	2172	LRTL2	0.851	0.655	0.389	0.876	0.641	0.056	0.866
	2534	LRTL3	0.657	0.318	0.815	0.968	0.972	0.606	0.651
	2388	LRTL4	0.800	0.396	0.713	0.905	0.920	0.589	0.810
5950	5950	SUMMER	0.862	0.806	0.793	0.895	0.631	0.432	0.618
	5950	FALL	0.907	0.804	0.785	0.805	0.579	0.302	0.903
	5950	WINTER	0.130	0.774	0.340	0.776	0.480	0.197	0.130
646	646	UA2	0.936	0.740	0.805	0.986	0.644	0.657	0.984
		Average	0.771	0.635	0.688	0.903	0.752	0.453	0.754

Table 3. 3 Overall F1 score comparisons of different methods (consider 15-frame acceptance of deviation from ground truth)

The recent rapid development of deep learning-based methods has shown better generalized features than that of conventional approaches. We also compared our proposed method with AlexNet conv3 layer feature-based approach [13], [35], [46], NetVLAD [2], [3], and CALC [36]. We used AlexNet conv3 layer features pre-trained on ImageNet [49] under the Caffe framework [46] to replace the convention features like low-resolution whole frame descriptor and HOG feature vector. We directly employed linear full search, i.e., single nearest neighbour, with the conv3 features and reported the best match pairs as the recognition results. For NetVLAD and CALC, we directly applied their trained models to the four challenging datasets and also adopted linear full search for the best match pair searching.

Table 3.3 shows the overall F1 scores of different methods on the four datasets under 15-frame acceptance of deviation from the ground truth. We can see that AlexNet conv3 approach gives the best F1 score, 0.903 on average. This is highly related to the generalization of deep features. However, the dimension of the AlexNet conv3 layer is 13 × 13 × 384, and the length of the feature vector is 64,896. The second highest average F1 score is given by our proposed FMPR, with is 0.771. By using our proposed key frame recognition for accumulated deviation elimination, we achieve the best and the second-best F1 scores in the testing sequences, “FALL” and “SUMMER”, 0.907 and 0.862 respectively. These two sequences

contain natural places as shown in Figure 3.18, frames captured on the outskirts are similar to each other and our fast reference frame tracking encounters ambiguities in decision making. Our proposed key frame recognition is useful to alleviate the ambiguities by re-setting the tracking tube to the location of the recognized key frame. In this work, we focus on fast visual place recognition for situations with varying speeds and changing lighting conditions but not extreme changes in appearance like the spring-winter changes. For such a reason, we did not prepare our work for seasonal changes and hence our performance for the "WINTER" testing sequence is less satisfactory.

NetVLAD and our FMPR without key frame recognition give similar overall performance on the four datasets in average, 0.752 and 0.754 F1 scores respectively. We believe that the performance of NetVLAD would be better if we fine-tune the model on similar datasets. Nevertheless, labelled training data is not always available, and it can be expensive for practical applications. The training of NetVLAD requires GPS stamps to construct a number of training pairs and it has already trained for place recognition. Therefore, we directly compared with the pre-trained model and we achieved comparable performance to them. The feature vector lengths of NetVLAD and ours without key frame recognition are 4,096 and 3072 (=64 × 48) respectively, which are 15.84 (=64,896/4,096) and 21.13 (=64,896/3,072) times shorter than that of the AlexNet conv3 vector. CALC also attains 0.453 F1 score which is a reasonable performance with a short vector length of 1,064.

For the conventional sequence-based approaches, SeqSLAM and ABLE-M have reasonable performance in average, 0.635 and 0.688 respectively. However, SeqSLAM cannot performance properly on LRTS and LRTL datasets which consist of varying speed situations. In addition, ABLE-M cannot work properly on nighttime testing sequences such as "LRTS2" and "LRTL2". The use of normalized low-resolution whole frame descriptor helps to deal with the nighttime testing sequences as it focuses on the structural features of a frame. SeqSLAM

and Ours use this descriptor to compute the recognition results such that both methods can achieve satisfactory performance on nighttime testing sequences.

Total no. of frames in database	Total no. of frames in testing seq.	Testing sequence	Ours (FMPR)	SeqSLAM [7],[42]	ABLE-M [31],[32]	AlexNet conv3 [13],[35],[46]	NetVLAD [2],[3]	CALC [36]	Ours (without Key Frame Recognition)
669	617	LRTS2	4.619	7.978	1.354	30.061	3.139	1.763	4.768
	634	LRTS3	5.237	7.473	1.342	30.352	3.270	1.780	5.110
	580	LRTS4	5.988	8.215	1.324	29.516	3.348	1.806	3.683
3169	2172	LRTL2	4.805	8.580	2.234	152.763	16.357	9.435	4.569
	2534	LRTL3	6.719	8.809	2.228	147.782	16.009	9.470	4.217
	2388	LRTL4	7.600	8.729	2.177	147.677	16.052	9.540	4.274
5950	5950	SUMMER	6.010	16.505	2.300	278.374	31.347	18.195	3.606
	5950	FALL	6.274	16.066	2.648	274.939	31.413	18.345	1.204
	5950	WINTER	4.342	15.931	2.792	272.965	30.520	18.175	1.374
646	646	UA2	6.709	7.086	1.318	29.031	3.056	1.749	1.915
		Average	5.830	10.537	1.972	139.346	15.451	9.026	3.472

Table 3. 4 Overall time cost comparisons of different methods (in millisecond, ms)

The time costs of all the comparing methods running by CPU in terms of millisecond are listed in Table 3.4. The CPU used for the comparison is Intel Core i7-4790@3.6GHz with 16GB memory. Note that the source codes of SeqSLAM and NetVLAD are written in MATLAB and the others are written in C++ without any parallel programming and code optimization. The feature extraction of the three CNN-based approaches, i.e., AlexNet conv3, NetVLAD and CALC, are written in Python and the extracted features are stored in text files when processing in C++. We divided the time costs reported by MATLAB by 10 for fair comparisons with C++. This is an assumption made in our previous work [50] based on extensive experiments on super-resolution using MATLAB and C++.

The average feature extraction time of a frame of AlexNet conv3, NetVLAD and CALC are 128.55, 168.93 and 22.4 ms repsectively. The processing time per frame of each method highly depends on the use of the searching technique and the vector length. It is obvious that AlexNet conv3 gets the largest time cost. It requires 267.896 ms per frame (=139.436+128.55) on average. Besides, NetVLAD (15.451 ms) and CALC (9.026 ms) offer reasonable searching time costs per frame but the feature extraction time of NetVLAD is long because of the employment of a deep network, VGG16 [41]. SeqSLAM is sensitive to the length of the

sequences as it uses linear full search, in which around 16 ms is required for long sequences and around 8 ms for short sequences. ABLE-M employs binary sequence codes to speed up the process so that they achieve a very small processing time cost, 1.972 ms per frame. For our proposed FMPR with and without the key frame recognition, we require 5.83 and 3.472 ms respectively. This means that our proposed two-stage key frame recognition strategy effectively simplifies the key frame recognition such that the overhead can be minimized. Compared to the best F1 approach, AlexNet conv3 which attains 0.903 F1 score. We achieve a comparable F1 score, 0.771 and is much faster than it by a factor of 45.95 times (=267.896/5.83).

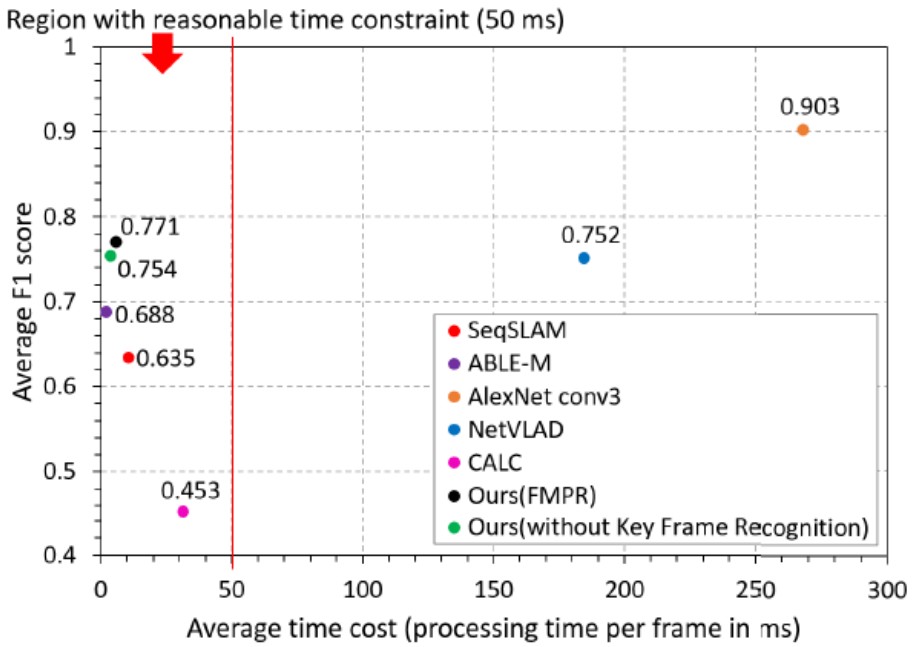


Figure 3. 30 Comparisons of the average F1 score and average time cost of our proposed FMPR with and without Key Frame Recognition and other approaches

Figure 3.30 shows the average F1 score versus the average time cost per frame of all methods. The best method should get both high F1 score and low average time cost. Therefore, the desirable locations are at the left-top region of Figure 3.30. Clearly, AlexNet conv3 approach (orange dot) is at the right-top corner which offers high F1 score but also high average time cost. Ours with and without Key Frame Recognition (green and black dots) achieve similar

performance compared to NetVLAD (blue dot) and are closer to the left-top region, i.e., lower average time cost. The left region of the red line in Figure 3.30 indicates the region with reasonable processing time, 50 ms, per frame for real-time applications under a certain time constraint. In this context, we fulfil the time constraint and is the best method to be selected. Hence, our FMPR offers a better balance between the accuracy and the processing time per frame, especially for real-time applications with battery-powered and resource-limited devices.

3.1.4 Conclusion on Conventional Learning Approach

Our work on conventional machine learning approach for place recognition has demonstrated the objective of offering fast algorithm for situations with varying speed situations and changing lighting environments. We give the recognition results using low-resolution whole frame descriptor tracking with the concept of tube of frames, for which both the estimation of the next best match based on previous results and the current feature matching of the current incoming frame are utilized. The idea of tube of frames emphasizes the linkage with the previous recognition results in which there are temporal logic constraints on the movement of the vehicle, and these can be used to narrow down the search space for the current pair matching process. For the problem of accumulated deviation due to fast reference frame tracking, an efficient two-stage key frame recognition is also introduced. When a key frame is confidently recognized, the corresponding location will be used to rectify the tracking tube location and hence the accumulated deviation can be cleared. Our proposed FMPR attains a high F1 score and is less sensitive to the length of sequence. FMPR is also a good alternative when the balance of performance and processing time per frame is important, especially for the applications for battery-powered and resource-limited devices. Readers may also note that visual place recognition does not mean to replace or even compete with other high-end sensor-based approaches or GNSS. Various techniques, including the visual place recognition

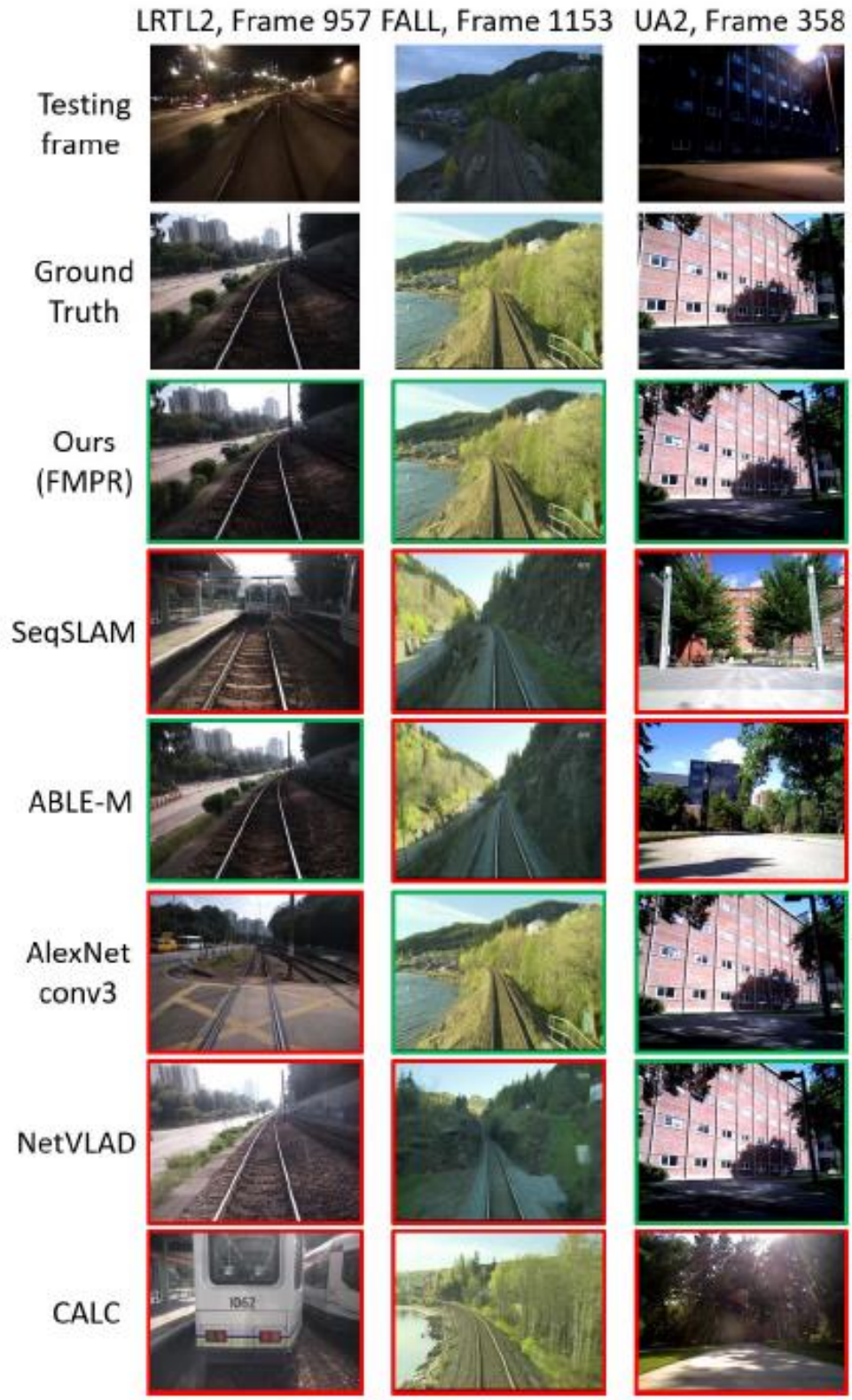


Figure 3. 31 Examples of match pairs from our proposed method (FMPR) and other approaches. There are three columns and one column for one example. The first and second rows show the testing frames and the ground truth for the corresponding example respectively. The third row displays the match pairs from our FMPR approach, the fourth row is the results from different approaches. The green-bounded match pairs indicate the correct match pairs under 15-frame acceptance of deviation from the ground truth while red-bounded match pairs mean incorrect match pairs

methods discussed in this work, can be integrated into a more comprehensive autonomous driving system.

In the next section, we will study different CNN model sizes to find out a model which is suitable for fast place recognition approach. The concept of tube of frames will also be combined with the lightweight CNN model for further improving the recognition performance.

3.2 Deep Learning Approach for Place Recognition

From our work on convention learning approach place recognition, we have shown that the temporal information, i.e., the idea of tube of frames, is useful for benefiting both the accuracy and efficiency of the visual place recognition system. Also, recent development of deep learning-based methods has demonstrated that deep features are more robust than traditional hand-crafted features. Nevertheless, Heavy computational cost (the feature extraction time) and difficulty in collecting a large amount of labelled training data are two well-known weaknesses of CNN-based approaches. To tackle these weaknesses, Merrill and Huang [36] proposed a shallow CNN model and an automatic training data generation method. The lightweight shallow CNN model offers faster feature extraction time by sacrificing the discrimination power of the input frames. In their training data generation, they applied a random perspective transformation to every input frame such that each input is automatically paired up with its self-perspective transformed version as the ground truth training pairs.

In this section, we discuss our work on deep learning-based approach for place recognition. Apart from taking the better generalization about the deep features for improving recognition performance, we also tackle the problem of unknown starting location of the vehicle by using the idea of tube of frames, i.e., the temporal correlation between consecutive frames. We further develop our tubing strategy in the form of dynamic tube sizes. We study

and improve the CNN model proposed in [36] by modifying the way of generating the training pairs with additional consideration to changes in appearance, illumination, and viewpoints.

3.2.1 Convolutional Autoencoder Network for Place Recognition

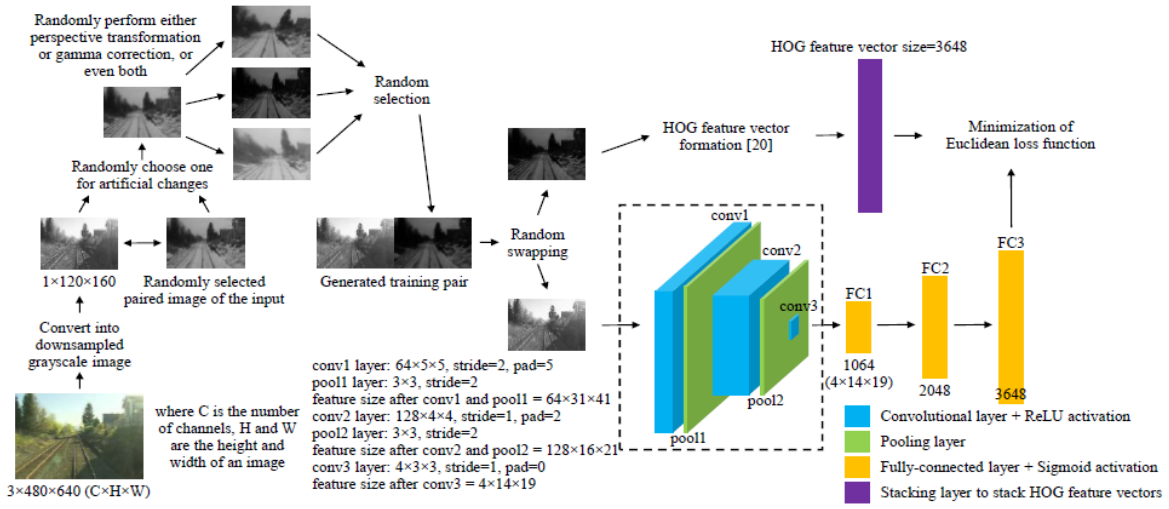


Figure 3. 32 Illustration of the CALC network architecture and its training process [36] with our proposed modifications in the training data generation. Note that the training pair covers the problems of changes in appearance, illumination, as well as viewpoints

Figure 3.32 shows the training process of the proposed model [36] with our modifications in the training data generation. We follow their training process[1] under the Caffe framework [46] to improve the model. Histograms of Oriented Gradients (HOG) [34], [45], [51], [52], is a well-known hand-engineered feature vector which is robust to changes in lighting conditions because of its local contrast normalization. [36] employed HOG for their network training for which it benefits from i) Smaller size of the extracted features ($\mathbb{R}^{3,648}$) compared to AlexNet conv3 features ($\mathbb{R}^{64,896}$), to achieve reasonable data compression; ii) Illumination invariance property of HOG to handle changes in lighting conditions; iii) Perspective transformed training data to further enhance the features for place recognition tasks. Their proposed network targets at reproducing the same HOG feature vector of an image pair

[1] Source code and pre-trained model are available online at https://github.com/rpng/calc. Please refer to it for the details.

based on an image pair which always represents the same place. For the loss function, they simply used Euclidean loss function, i.e., $L2$ norm [45] to minimize the different between the HOG feature vector and the deep feature vector given the network. For the online practicing stage, only the convolutional layers are kept, i.e., the dashed box in Figure 3.32, for further speeding up the feature extraction stage. Therefore, the network learns to map an input image ($\mathbb{R}^{120\times160}$) to a feature space ($\mathbb{R}^{1,064}$). Note that the input image size is 120 × 160 while the output feature size is 1,064.

Our proposed improvement in the training data generation is that we insert more variations in the paired training data. Apart from the random perspective transformation, we also include a random gamma correction to the input images and a random selection of the corresponding image for the data generation. The formula for the gamma correction is shown in Eq. 3.26 and γ is randomly chosen from 0.1 to 2.5.

$$I_c = (\frac{I}{255})^{\gamma} \times 255 \tag{3.26}$$

where I is the input image, I_c is the gamma corrected image and γ is the gamma used to correct the brightness of the input image via using non-linear mapping of pixel values. If $\gamma < 1$, the corrected image will be darker than the original input image. For $\gamma > 1$, the opposite observation is made. We are not restricted to use the input image to generate the ground truth label for training. Our proposed procedure for generating a training pair is as follows (also refer to the left-hand side of Figure 3.32).

(i) We randomly select one of the paired images of the input from the dataset or use the input image directly as an image pair. For example, if a place has been recorded in 4 different time slots in the dataset, we will have possible 4 image pairs of this place. We randomly pick one of them and use it to create a training image pair. Note that all the input images to the model are with size of 120 × 160 and is grayscale.

(ii) From each training image pair, we randomly choose one image to perform either the perspective transformation or the gamma correction (Eq. 3.26), or even both perspective transformation and gamma correction.

With our data generation strategy, we can then include the three practical problems to generate the training pairs, i.e., changes in appearance, illumination as well as perspective.

3.2.2 Temporal Correlation based Initialization

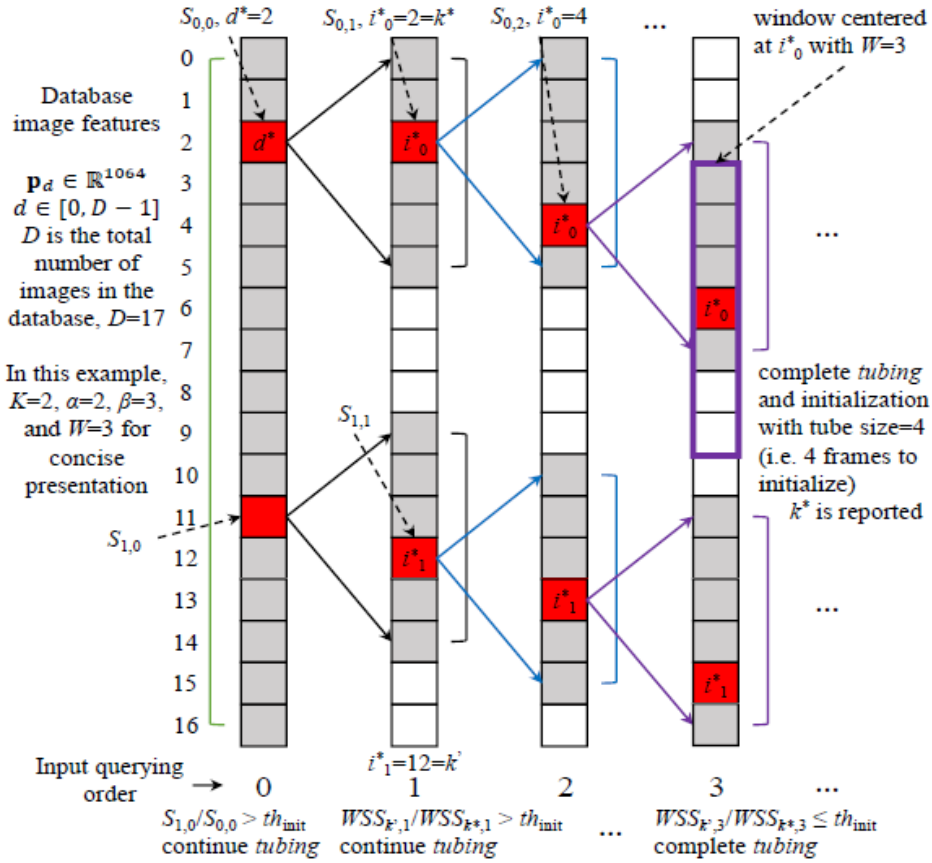


Figure 3. 33 Our proposed temporal correlation-based initialization stage

Our previous work on conventional learning approach for visual place recognition [34], [51] found that only datasets for situations with constant speed can benefit from long image sequences. However, practical situations with varying speeds are burdened with the excessive consideration to the historical information, i.e., a group of shorter image sequences is more suitable for sequences with varying speeds. An important question is that what is a suitable size

of image sequences? This means that how much temporal correlation we have to consider so as to benefit from using the historical information. The concept of dynamic tube sizes or tubing is introduced to address the problem. We propose a weighted sum of the searching similarity scores (*S*) of consecutive frames, and the weight of each searching similarity score is depending on the differences between the current querying frame and the previous frames.

Figure 3.33 shows the pipeline of our proposed temporal correlation-based initialization graphically. For the 1st query frame (the most left column), we perform the linear full search to find the match pair as shown in Eq 3.27.

$$d^* = \underset{d\in[0,D-1]}{\arg\max}\, C(\mathbf{q}_t, \mathbf{p}_d) = \underset{d\in[0,D-1]}{\arg\max}(\mathbf{q}_t \cdot \mathbf{p}_d) \tag{3.27}$$

where $C(\mathbf{q},\mathbf{p})$ is the Cosine similarity defined as the cosine of the angle difference between normalized vectors **q** and **p**. In this work, **p** and **q** represent the normalized deep feature vectors of the database images and query images respectively extracted by the network model in section 3.2.1. *D* is the total number of frames in the database, *t* is the input query order, $t = 0$ for the 1st query frame. Neighbour d^* is the single nearest neighbour to the 1st query frame among the database with the highest similarity score, $S_{0,t=0} = C(\mathbf{q}_t, \mathbf{p}_{d^*})$ and d^* is regarded as the match pair for output. If the ratio of $S_{0,t=0}$ to $S_{1,t=0}$ (the second highest similarity score found outside the window centered at d^*, with window size *W*) is smaller than a threshold, th_{init}, the confidence of this match pair is high and the initialization can be done with only the 1st query frame. Otherwise, we record the top *T*% of the nearest neighbours with the similarity scores denoted as S_k, $k \in [0,K\text{-}1]$ and $K=D \times T\%$. The search range of the next query frame is based on the *K* nearest neighbours and i_k denote the location of the k^{th} neighbour in the database. Starting from the 2nd query frame, the weighted similarity score is computed by Eqs. 3.28 and 3.29.

$$i_k^* = \underset{i \in [i_k - \alpha, i_k + \beta]}{\arg\max} C(\mathbf{q}_t, \mathbf{p}_i) \tag{3.28}$$

$$S_{k,t} = C(\mathbf{q}_t, \mathbf{p}_{i_k^*}) \times (1 - C(\mathbf{q}_t, \mathbf{q}_{t-1})) \tag{3.29}$$

where α and β are the upward and downward search offset respectively. For each i_k, it has its own search range, and we find the corresponding single nearest neighbour i^*_k. If $i^*_k - \alpha < 0$, we start to search from the 1st database frame; If $i^*_k - \beta \geq D$, we stop the search process once reaching the last database frame. $S_{k,t}$ is the weighted similarity score of the k^{th} neighbour in time t. We weight the score using the difference between the current and previous query frames. If the two frames are very similar, this means that we can neglect the current score as there is very little new information given by the current query frame. Note that i_k is updated for each query frame based on i^*_k, hence we only keep the K nearest neighbours to each query frame. We report the match pair of the current query frame (k^*) using Eq. 3.30.

$$k^* = \underset{k \in [0, K-1]}{\arg\max} (S_{k,t} + \sum_{a=0}^{t-1} S_{k,a} \times C(\mathbf{q}_t, \mathbf{p}_a)) \tag{3.30}$$

$$WSS_{k^*,t} = S_{k^*,t} + \sum_{a=0}^{t-1} S_{k^*,a} \times C(\mathbf{q}_t, \mathbf{p}_a) \tag{3.31}$$

if t=1, $S_{k,0}$ is obtained from the linear full search of the 1st query frame, i.e., $S_{k,t=0} = \{S_{0,0}, S_{1,0}, \ldots, S_{K-1,0}\}$. When the current query frame is different from the past query frames, the vehicle goes far away from the past locations and the influence of the historical information on the current decision making should be diminished. The weighted sum of scores of location k^* for the current query frame is calculated using Eq. 3.31. Similar to the case of the 1st query frame, we compute the ratio of $WSS_{k^*,t}$ to $WSS_{k^*,t}$ (k' is the location where has the second highest weighted sum of scores found outside the window centered at k^*, with window size W, the purple box in Figure 3.33). If the ratio is smaller th_{init}, high confidence of k^* is observed and the initialization is done with tubing.

3.2.2.1 Experimental Results and Analysis

3.2.2.1.1 Dataset for Fine-tuning and Module Parameters

We used parts of the Nordland dataset [42] and removed the tunnel and stop frames for the network fine-tuning. This dataset contains 4 long rail sequences recorded at 4 seasons and it has been time-synchronized, hence any frame in one of the sequences represents the same frame in the other three sequences. We used the first 10,000 frames of each sequence. After the removal of stop and tunnel frames, there are 7,705 frames for each sequence and 30,820 (=7,705 × 4) frames in total for the fine-tuning.

We considered the top 10% (T% = 10%) of the nearest neighbours for the initialization. α and β are set to 5 and 10 respectively. The window size W is set to 3 and th_{init} is predefined as 0.8.

3.2.2.1.2 Datasets for Comparisons

In our experiments, 3 challenging datasets were included, and we compared to several state-of-the-art approaches, CALC [36] and AlexNet conv3 deep feature-based approaches [13], [35], [53]. CALC is the network model that we fine-tuned for our proposed method. Note that CALC merely focuses on the discrimination power of its deep features, they applied the simplest single nearest neighbour search to find the match pair. For AlexNet conv3 deep feature-based approaches, we directly extract the conv3 features from two AlexNets pre-trained on two datasets, ImageNet [35] and Places365 [53]. The former one is for object classification tasks and the latter one is for single scene recognition tasks. These approaches also use the single nearest neighbour search.

3.2.2.1.2.1 Alderley Dataset

This dataset is introduced in [7] which focuses on extreme changes in weather and lighting conditions. We extracted the first 2,000 frames of the daytime sequence to build the database and there are 2,069 frames of the nighttime sequence correspond to these 2,000

database images. Note that the ground truth of this dataset is very close to a diagonal line which means constant speed situation is considered.

3.2.2.1.2.2 Nordland Dataset

We used the last 5,000 frames of each sequence for comparisons. We made sure that there is no overlap with the training data, and we did not remove the tunnel and stop frames for the comparisons. It is because this is real situation for practical applications, and this is for testing whether the methods can perform well with slow moving or even stop frames. The database was formed using the "Spring" sequence. For examples of this dataset, please refer to Figure 3.18 in section 3.1.3.1.2.

3.2.2.1.2.3 Light Rail Transit (LRT) Dataset

LRT dataset was captured directly from a public transportation system in Hong Kong. There are 4 sequences of the same route, 3 in the daytime and 1 at the nighttime. The dataset consists of many practical difficulties like varying speeds, extreme changes in lighting environments, and motion blurring. On average, there are 2,566 frames for a sequence in this dataset. We used one of the daytime sequences to build the database. As the sequences are not time-synchronized, we manually marked the ground truth of all the sequences. Please refer to Figure 3.17 in section 3.1.3.1.1.

3.2.2.1.3 High Confident Initialization with Temporal Module

	Fine-tuned CALC									
	Without tubing strategy		With tubing strategy		CALC [36]		AlexNet, Places365 [35]		AlexNet, ImageNet [53]	
querying sequence	Precision	Average tube size	Precision	Average tube size	Precision	Average tube size	Precision	Average tube size	Precision	Average tube size
Alderley	0.2	1.0	**0.5**	11.1	0.1	1.0	0.0	1.0	0.1	1.0
Summer	0.7		**1.0**	8.0	0.5		0.5		0.5	
Fall	0.9	1.0	**1.0**	2.5	0.8	1.0	0.6	1.0	0.3	1.0
Winter	0.4		**0.7**	7.6	0.3		0.2		0.0	
LRT2	0.6		**0.7**	54.1	0.2		0.6		0.2	
LRT3	0.6	1.0	**1.0**	80.4	0.6	1.0	0.7	1.0	0.6	1.0
LRT4	0.6		**0.7**	27.4	0.5		**0.7**		0.6	
Average	0.571	1.0	**0.800**	27.3	0.429	1.0	0.471	1.0	0.329	1.0

Table 3. 5 The initialization performance of various approaches (the best scores are in **bold** typeface)

We show the advantage of using our proposed tubing strategy for the initialization to localize the starting location of the vehicle. For computing the precision, a match pair is regarded as correct if its difference between the ground truth is fewer than 5 frames. We randomly selected 10 starting points for each querying sequence and the precisions of the initialization of various methods are listed in Table 3.5.

Without the proposed tubing strategy, the initialization of the fine-tuned CALC is simply done with the single nearest neighbour search. Therefore, the average tube size is always 1.0 as no temporal correlation between consecutive query frames is included. It is obvious that the initialization performance is boosted with the tubing strategy. On average, we get 0.229, = 0.800 – 0.571 increase in precision. Note that high confident initialization is always the first step in comprehensive localization/navigation systems. Compared to the original CALC, we also show our improvement with our modified training data generation method. Considering only the discrimination power of the deep features, the original CALC got 0.429 in average precision while our fine-tuned CALC attained 0.571. We can also see that the

AlexNet pre-trained on Places365 outperformed the AlexNet pre-trained on ImageNet in our experiments. On average, these two methods attained 0.471 and 0.329 precision respectively. This implies that one can benefit from the model pre-trained on task-related datasets. We offered 0.4, = 0.800 – (0.471 + 0.329) / 2, improvement in precision compared to that of the two AlexNet conv3 deep feature-based approaches.

More importantly, our tubing strategy dynamically groups a number of query frames to make a confident initialization decision. We have had larger tube size for LRT2 and LRT3 sequences, 54.1 and 80.4 respectively. This is due to the fact that we have to handle situations with varying speeds. The random selected starting points sometimes locate at stop-frame locations, and this requires more frames to make the decision as stop frames offer very little or even no new information. Hence, our tubing strategy needs more frames to make a confident decision. On average, the tubing strategy requires 27.3 frames to localize the starting locations of the vehicle which costs around 1.1 second in a 25-fps system. This means that 1.1 second is needed to initialize the starting location of the vehicle in practice.

3.2.3 Efficient Place Recognition Strategy

After studying the high confident initialization strategy, we assume that the starting position of a vehicle can be known, and we further study an efficient place recognition strategy to reduce the search range of each new coming frame.

Once a confident initialization is done and the starting location is localized, we can further reduce the search space for the coming query frames since the vehicle must travel along a route gradually without any sudden jump from one point to another point. Therefore, linear full search is not necessary, and the reduced search space helps to improve both the place recognition efficiency and accuracy.

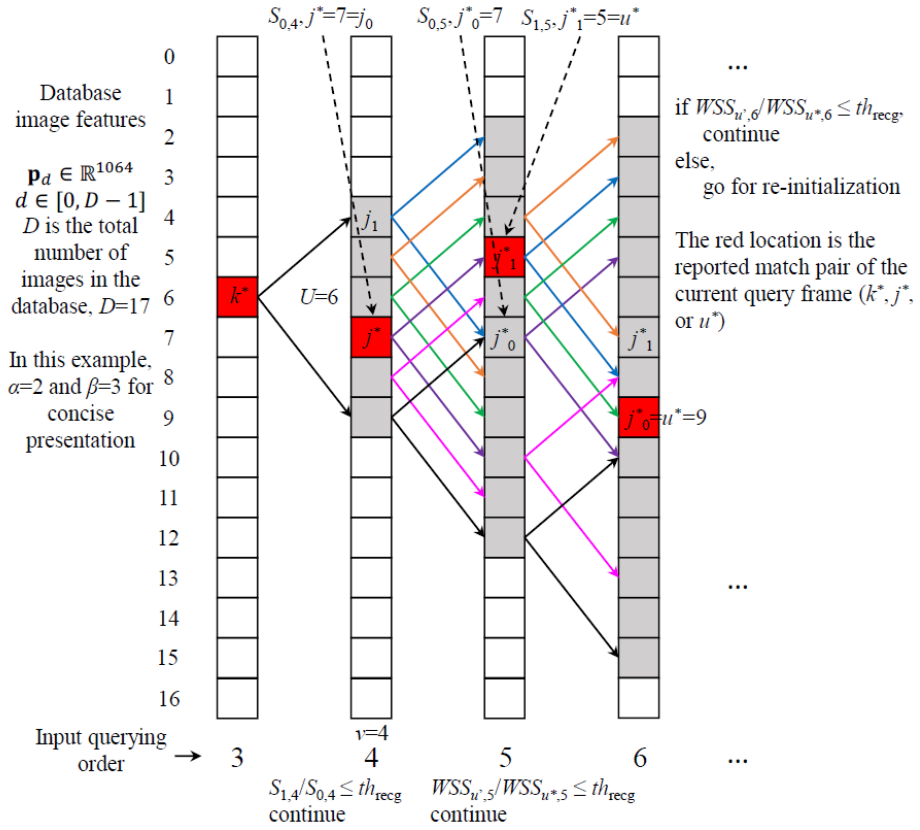


Figure 3. 34 A graphical illustration of our proposed efficient place recognition strategy

Figure 3.34 shows our proposed efficient place recognition strategy graphically based on the initial match pair, k^* (the left most column), which comes from our tubing initialization. For the 1st query frame after obtaining k^*, we perform a linear full search in a new reduced search space to find the match pair using Eq 3.32.

$$j^* = \underset{j \in [k^*-\alpha, k^*+\beta]}{\arg\max} C(\mathbf{q}_t, \mathbf{p}_j) = \underset{j \in [k^*-\alpha, k^*+\beta]}{\arg\max} (\mathbf{q}_t \cdot \mathbf{p}_j) \qquad (3.32)$$

where Cosine similarity is used, $C(\mathbf{q}, \mathbf{p})$, for the similarity measure. $(\mathbf{q} \cdot \mathbf{p})$ is defined as the cosine of the angle difference between two normalized vectors $\mathbf{p}$ and $\mathbf{q}$. $\mathbf{p}$ and $\mathbf{q}$ denote the normalized ConvNet feature vectors of the database frame and query frame given by the network model mentioned in section 3.2.1, refer to Figure 3.32, respectively. t is the current querying order. j^* is the single nearest neighbor to the current query frame among the new reduced search space with the highest similarity score, $S_{0,t} = C(\mathbf{q}_t, \mathbf{p}_{j^*})$. If the ratio of $S_{0,t}$ to $S_{1,t}$

(the second highest similarity score found in the search space) is smaller than a threshold, th_{recg}, the confidence of j^* is at a satisfactory level and we will keep the tubing to continue using the temporal correlation between consecutive query frames. We believe that once we confidently localize the vehicle, fast recognition or tracking can be performed to enhance the efficiency of the model until the confidence level of the output drops below a certain threshold value. If the ratio is larger than th_{recg}, we will perform re-initialization. After finding j^*, all the similarity scores in the search space are denoted as $S_{u,t}$, $u \in [0, U\text{-}1]$ where $U=\alpha+\gamma+1$, and j_u denotes the location u^{th} neighbor in the search space which also defines the search space for the next query frame. From the next query frame, the weighted similarity score is computed using Eqs. 3.33 and 3.34.

$$j_u^* = \underset{j \in [j_u-\alpha, j_u+\beta]}{\arg\max}\, C(\mathbf{q}_t, \mathbf{p}_j) \tag{3.33}$$

$$S_{u,t} = C(\mathbf{q}_t, \mathbf{p}_{j_u^*}) \times \left(1 - C(\mathbf{q}_t, \mathbf{p}_{t-1})\right) \tag{3.34}$$

where α and γ have been defined previously. Each j_u has its own search space and the corresponding single nearest neighbor is denoted as j^*_u. The weighted similarity score of the u^{th} neighbor in time t is denoted as $S_{u,t}$. Note that only the U nearest neighbors are kept for decision making and j_u is updated continuously according to j^*_u. The match pair of the current query frame (u^*) is given by Eq. 3.35.

$$u^* = \underset{u \in [0, U-1]}{\arg\max}(S_{u,t} + \textstyle\sum_{a=v}^{t-1} S_{u,a} \times C(\mathbf{q}_t, \mathbf{p}_a)) \tag{3.35}$$

$$WSS_{u^*,t} = S_{u^*,t} + \textstyle\sum_{a=v}^{t-1} S_{u^*,a} \times C(\mathbf{q}_t, \mathbf{p}_a) \tag{3.36}$$

where the tube is reset after the initialization and v is the querying order in which j^* is computed. $S_{u,v}$ is calculated during the computation of j^* mentioned in above. Eq. 3.36 shows the computation of the weighted sum of scores of location u^* for the current query frame. Similarly, if the ratio of $WSS_{u^*,t}$ to $WSS_{u',t}$ (the second highest weighted sum of scores found in

the current search space) is smaller than a threshold, th_{recg}, the confidence of u^* still maintains at a satisfactory level and u^* is reported as the match pair for output. Otherwise, re-initialization will be activated.

3.2.3.1 Experimental Results and Analysis

3.2.3.1.1 Dataset for Fine-tuning and Module Parameters

Same as mentioned in section 3.2.2.1.1, we used parts of the Nordland dataset [42] and removed the tunnel and stop frames for the network fine-tuning. This dataset contains 4 long rail sequences recorded at 4 seasons and it has been time-synchronized, hence any frame in one of the sequences represents the same frame in the other three sequences. We used the first 10,000 frames of each sequence. After the removal of stop and tunnel frames, there are 7,705 frames for each sequence and 30,820 (=7,705 × 4) frames in total for the fine-tuning.

For the parameters of our proposed efficient place recognition strategy, α and β are set to 5 and 10 respectively, and th_{recg} is pre-defined as 0.75.

3.2.3.1.2 Datasets for Comparisons

Three challenging datasets were included in this work, and we compared to several state-of-the-art approaches, namely SeqSLAM [7], ABLE-M [31], [32], CALC [36] and AlexNet conv3 deep feature-based approaches [13], [35], [53]. SeqSLAM utilizes down-sampled grayscale normalized images as features for pair searching and assumes constant speed situation. ABLE-M is also a sequence-based approach in which different groups of consecutive images are represented by different binary sequence codes for pair searching. We denote ABLE-M using different sequence lengths as ABLE-M l where l is the size of an image sequence, l=1, 150 and 300 in our experiments. CALC is the model that we fine-tuned for our proposed method. CALC relies on the discrimination power of its deep features, they applied the simplest single nearest neighbour search for pair searching. For AlexNet conv3 deep feature-based approaches, we directly extract the conv3 features from two AlexNets pre-trained

on two datasets, ImageNet [35] and Places365 [53]. The former one is for object classification tasks and the latter one is for single scene recognition tasks. These approaches also use the single nearest neighbour search.

3.2.3.1.2.1 UA Dataset

The UA dataset has introduced in [43]. This dataset focuses on changes in lighting conditions and has been time-synchronized. We extracted two sequences from this dataset, one daytime and one nighttime, both have 646 frames. The daytime sequence was used to construct the database, please refer to Figure 3.20 for the examples of this dataset.

3.2.3.1.2.2 Nordland Dataset

We used the last 5,000 frames of each sequence for comparisons. We made sure that there is no overlap with the training data, and we did not remove the tunnel and stop frames for the comparisons. It is because this is real situation for practical applications, and this is for testing whether the methods can perform well with slow moving or even stop frames. The database was formed using the "Spring" sequence. For examples of this dataset, please refer to Figure 3.18 in section 3.1.3.1.2.

3.2.3.1.2.3 Light Rail Transit (LRT) Dataset

LRT dataset was captured directly from a public transportation system in Hong Kong. There are 4 sequences of the same route, 3 in the daytime and 1 at the nighttime. The dataset consists of many practical difficulties like varying speeds, extreme changes in lighting environments, and motion blurring. On average, there are 2,566 frames for a sequence in this dataset. We used one of the daytime sequences to build the database. As the sequences are not time-synchronized, we manually marked the ground truth of all the sequences. Please refer to Figure 3.17 in section 3.1.3.1.1.

3.2.3.1.3 Evaluation Metrics

To evaluate different approaches, we sorted all match pairs from different approaches based on their weighted similarity scores. Generally, a match pair with high score is more likely to be correct. For computing the precision, a pair is regarded as correct if its difference from the ground truth is less than 5 frames. We apply a set of 100 recall rates, 0.01 to 1.00 with each step = 0.01, to the sorted scores and generate the corresponding set of precisions and recall rates. We used F1 score to evaluate various approaches for concise comparisons which is computed by Eq. 3.25,

$$F1 = 2 \times \frac{P \cdot R}{P + R} \qquad \text{(Same as Eq. 3.25)}$$

where P is the precision defined as the ratio of the number of correct match pairs to the number of recalled match pairs. R is the recall rate [0.01, 1.00] and is defined as the ratio of the number of recalled match pairs to all the query frames. Note that high $F1$ is attained if and only if both P and R are high. Therefore, $F1$ can reflect the practicability of a method.

3.2.3.1.4 F1 Score Comparisons

querying sequence	Ours*	Ours	CALC [36]	AlexNet, Places365[53]	AlexNet, ImageNet[35]	SeqSLAM [7]	Downsampled image[7]	ABLE-M 300[31],[32]	ABLE-M 150[31],[32]	ABLE-M 1[31],[32]
UA	**0.941**	0.889	0.369	0.419	0.582	0.654	0.155	*0.914*	0.899	0.476
Summer	0.760	0.637	0.478	0.492	0.381	0.832	0.094	**0.952**	*0.932*	0.450
Fall	0.827	0.761	0.586	0.557	0.534	0.855	0.122	**0.961**	*0.944*	0.579
Winter	0.590	0.478	0.308	0.494	0.152	**0.741**	0.052	*0.695*	0.540	0.054
LRT2	0.505	0.522	0.319	*0.596*	0.297	**0.658**	0.144	0.098	0.219	0.068
LRT3	0.742	0.736	0.721	**0.843**	*0.775*	0.420	0.113	0.213	0.338	0.340
LRT4	0.700	0.743	0.687	**0.821**	*0.751*	0.445	0.226	0.249	0.332	0.297
Average	**0.724**	*0.681*	0.495	0.603	0.496	*0.658*	0.129	0.583	0.601	0.323

Table 3. 6 Overall F1 score comparisons of various approaches (the best scores are in **bold** typeface; the second-best scores are in *italics* typeface)

The overall F1 scores of different methods are listed in Table 3.6. For each approach, each pair of precision and recall rate gives a F1 score and here we compute the maximum F1 score of each approach. "Ours*" represents the method of the fine-tuned CALC with our modified training data generation method and the proposed tubing strategy while "Ours" represents the fine-tuned network model without the proposed efficient tubing strategy for

recognition. On UA dataset, “Our*” obtains a 0.572, = 0.941 – 0.369, F1 score improvement over the original CALC. On average, we have the highest F1 score, 0.724, throughout the three challenging datasets. This means that our proposed efficient recognition method has better adaptability to general situations. For the discrimination power and generalization about the ConvNet features. “Ours” gives better performance (0.681), compared to the original CALC (0.495), AlexNet Places365 (0.603), and AlexNet ImageNet (0.496) on the three datasets. This also reflects that the AlexNet pre-trained on the recognition-centric dataset, i.e., Places365, performs better than that of the classification-centric, i.e., ImageNet, dataset.

Apart from the deep learning-based methods, conventional sequence-based methods, namely SeqSLAM and ABLE-M have good performance on long sequences for situations with constant speed, the Nordland dataset. ABLE-M 300 (0.869 = (0.952 + 0.961 + 0.695) / 3) clearly outperforms ABLE-M 150 (0.805 = (0.932 + 0.944 + 0.540) / 3) on the Nordland dataset and provides evidence that sequences with constant speed benefit from large tube size. Nevertheless, the performance of these sequence-based methods drops drastically on the LRT dataset (0.56 = (0.098 + 0.213 + 0.249) / 3). This also gives evidence that large tube size can worsen the performance in varying speed situations. Therefore, dynamic tubing strategy is a must to get satisfactory performance in both situations with varying speeds and constant speed.

3.2.3.1.5 Time Cost Comparisons

Dataset	Ours*	Ours, CALC [36]	AlexNet [35],[53]	SeqSLAM [7]	ABLE-M [31],[32]
UA	49.6	67.0	288.4	2.0	0.6
Nordland	68.2	80.5	385.6	12.7	1.5
LRT	58.8	72.2	354.4	7.1	1.0
Average	58.9	73.2	342.8	7.3	1.0

Table 3. 7 Overall time cost comparisons of various approaches (millisecond, ms, only CPU is used)

The time cost comparisons of all the methods, including our approach and conventional approaches, SeqSLAM and ABLE-M are listed in Table 3.7 in terms of milliseconds. The CPU used for the time cost comparisons is Intel Core i7-6900k @ 3.2 GHz. The codes of SeqSLAM were written in MATLAB while ABLE-M and our efficient tubing strategy were written in C++. Therefore, we divide the time reported by MATLAB by 10 for a possible fair comparison. "Ours" and the original CALC should have the same time cost as both use the single nearest neighbour search and have the same network model. The two AlexNets pre-trained on two datasets should also have the same time cost.

Our first observation is that conventional approaches are much faster than the ConvNet feature-based approaches as there is no heavy computation of feature extraction by means of convolutions. Nevertheless, the discrimination power of ConvNet features obviously outperforms the traditional hand-engineered features like down-sampled grayscale normalized image and binary descriptor as shown in Table 3.6. SeqSLAM is sensitive to the length of sequence can also be reflected in Table 3.7. For the Nordland dataset which contains 5,000 frames, SeqSLAM requires 12.7 ms per frame while SeqSLAM only takes 2.0 ms per frame on the UA dataset which has 646 frames. For this reason, ABLE-M adopts image sequences to form binary sequence codes for efficient pair matching via the use of Hamming distance. Therefore, ABLE-M is the fastest method among all the methods which only takes 1.0 ms per frame on average.

For the ConvNet feature-based methods, "Ours*" is the fastest method because of the use of temporal correlation between consecutive query frames for the search space reduction. The time cost of the feature extraction of CALC model is 46.5 ms. This means that pair searching with our proposed tubing strategy costs only 12.4 ms, = 58.9 – 46.5 ms, on average. Compared to the linear full search method, 73.2 – 46.5 = 26.7 ms, our proposed tubing strategy is faster than it by a factor of 2.15. For the AlexNet conv3 feature-based approaches, on average

342.8 ms is required to match a frame because of the slow feature extraction stage and high dimensional feature vectors.

3.2.4 Conclusion on Deep Learning Approach

We have further developed the concept of tube of frames in our work on deep learning-based approach for place recognition. We study a lightweight CNN model for efficient place recognition and improve an automatic training data generation module to ease the burden of collecting a large amount of labelled training data. In our enhanced training data generation module, variations in appearance, seasons, lighting conditions, and perspectives are also considered for better generalization of extracted deep features. We also tackle the problem of unknown initial location of a vehicle with our novel dynamic tubing strategy. By making use of the temporal correlation between consecutive query frames, a high confident initialization with 80% accuracy can be attained compared to 57.1% without the tubing strategy, 47.1% and 32.9% of two direct AlexNet conv3 feature-based approaches.

We further extend the initialization module to an efficient place recognition method by proposing an efficient tubing strategy for recognition after the initialization. Similar idea of using temporal correlation with a sequence of consecutive query frames is applied to recognition once the initial location of a vehicle is found. We suggest using the weighted sums of the similarity scores based on the comparison of consecutive frames to obtain the final match pairs. The search space of the coming query frame is reduced and defined by the previous match pairs, hence efficient searching can be achieved. Our experimental results have demonstrated that the proposed efficient place recognition method gives a satisfactory $F1$ score of 0.724 compared to the second best (0.658, conventional sequence-based method) and the third best (0.603, AlexNet deep feature-based method). Our tubing strategy is also faster than the commonly used linear full search strategy be a factor 2.15, 12.4 ms versus 26.7 ms using a standard CPU device.

For the discriminative power of the deep features, we found that the feature extraction is sensitive to the input images, especially for the images with dynamic and unwanted objects like pedestrians and moving vehicles. This observation inspires us to make a study of quality scene reconstruction, in which we target at clean images for place recognition. It is well-known that clean images are not possible for real-life place recognition tasks. In the next chapter, we discuss our study of deep learning-based image inpainting, a useful technique for unwanted object removal.

3.3 Chapter Summary

In Chapter 3, we covered our work on both conventional machine learning and recent deep learning based visual place recognition. We target at efficient methods of visual place recognition such that the proposed methods could be further developed into real-time real-world applications.

For the conventional machine learning-based visual place recognition, we define key frames in a sequence as the places where are visually different from other frames in the same sequence and hence can be easily recognized. We analyse and represent key frames by few but effective feature patches with varying patch sizes. To improve efficiency in providing recognition results, we propose an efficient frame tracking module and a two-stage key frame recognition module. We predict the incoming match and define its search range based on the previous recognition results. The accumulated errors from the prediction are removed when a key frame is confidently matched. The two-stage key frame recognition module maintains the efficiency by only activating the detail feature patch matching stage when necessary. Our experimental results demonstrate that our proposed method can offer a better balance of the recognition performance and computational cost compared to both state-of-the-art conventional sequence-based methods and recent deep feature-based methods.

For the deep learning-based visual place recognition, we take the advantages of both the discriminative power of deep features and the use of temporal information given from the previous recognition results and query frames. We study a lightweight CNN model to minimize the feature extraction time cost of extracting deep features. We also improve an automatic training pair data generation module by adding more variations in self-augmented data including changes in lighting conditions, appearance, and viewpoints. By doing this, the trained model can extract features that are more robust to various types of changes encountered in practical situations. For the use of temporal information, we propose our dynamic tubing strategy for which it adaptively takes a group of previous frames into account based on the amount of new information given by the previous frames, i.e., the differences between the current query frame and previous frames. For example, if a vehicle stops, more frames need to be considered so as to have sufficient information for making a confident recognition. We propose a high confidence initialization module and an efficient recognition module based on the lightweight CNN model and our tubing strategy. Note that a high confidence initialization is important especially for the task of unknown starting location. The experimental results show that our improved automatic training pair generation module can obviously enhance the discriminative power of the extracted deep features compared to the existing CNN models. With our tubing strategy, we significantly boost the recognition performance compared to both conventional sequence-based methods and deep learning-based methods without using the temporal information.

To further improve recognition performance, one promising way is to study the robustness of the extracted deep features to various situations. We believe that extracting features from clean images can benefit the recognition performance. Clean images are regarded as images without unwanted objects such that dead cars, moving vehicles and pedestrians. This inspires our study on object removal via deep learning-based image inpainting.

Chapter 4 Learning Approach for Quality Scene Recognition

4.1 Deep Generative Image Inpainting Network

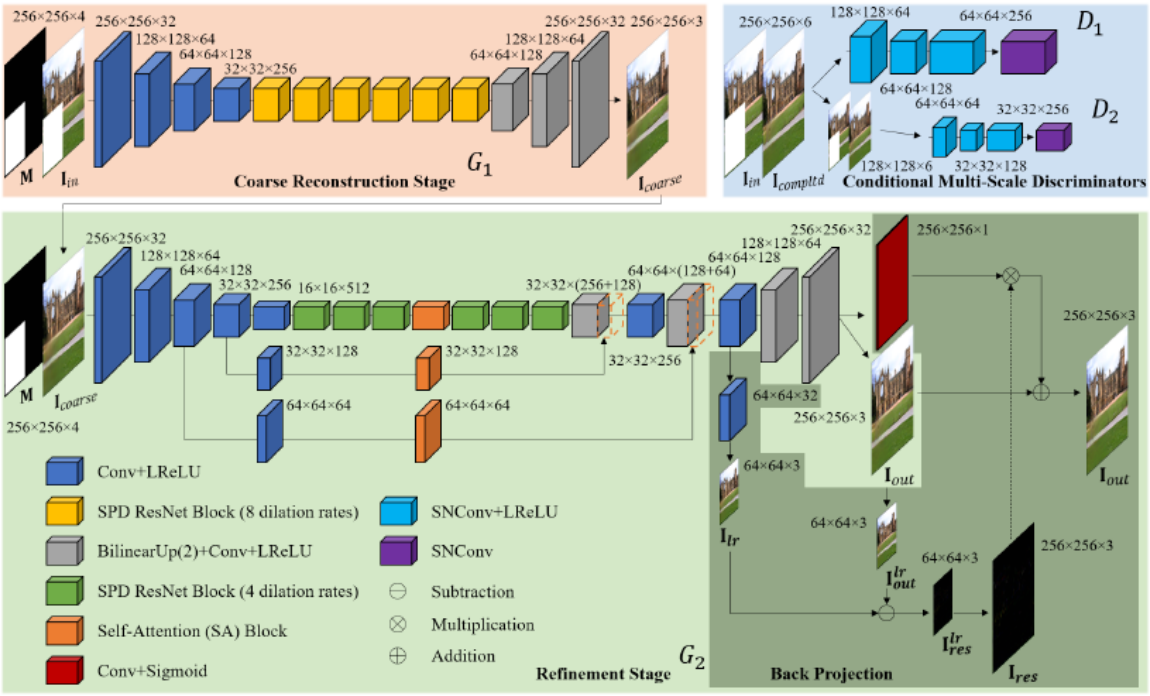


Figure 4. 1 Architecture of our proposed model for image inpainting. Our proposed model consists of two generators and two discriminators. The coarse generator G_1 at Coarse Reconstruction Stage and the second refinement generator G_2 at Refinement Stage constitute our DeepGIN which is used in both training and testing. The two discriminators D_1 and D_2 located within Conditional Multi-Scale Discriminators area are only used in training as an auxiliary network for generative adversarial training

Existing image inpainting approaches usually have certain assumptions of the shapes and sizes of the masked areas. We propose a Deep Generative Inpainting Network, named DeepGIN, to handle various types of wild masked images. To deal with different types of masks with various shapes and sizes, we propose a Spatial Pyramid Dilation (SPD) block for ensuring the model can capture the global semantics of various masked images.

Our proposed Deep Generative Inpainting Network (DeepGIN) consists of two stages as shown in Figure 4.1, a coarse reconstruction stage and a refinement stage. The first coarse generator $G_1(\mathbf{I}_{in}, \mathbf{M})$ is trained to roughly reconstruct the masked regions and gives $\mathbf{I}_{coarse}$. The second

refinement generator $G_2(\mathbf{I}_{coarse}, \mathbf{M})$ is trained to exquisitely decorate the coarse prediction with details and textures, and eventually forms the completed image $\mathbf{I}_{out}$ ($\mathbf{I}_{compltd}$). For our discriminators, motivated by SN-GANs [62], [83] and multi-scale discriminators [56], [57], we modify and employ two SN-GAN based discriminators $D(\mathbf{I}_{in}, \mathbf{I}_{compltd})$ which operate at two image scales, 256 × 256 and 128 × 128 respectively, to encourage better details and textures of local reconstructed patterns at different scales. Details of our network architecture and learning are shown below.

4.1.1 Network Design

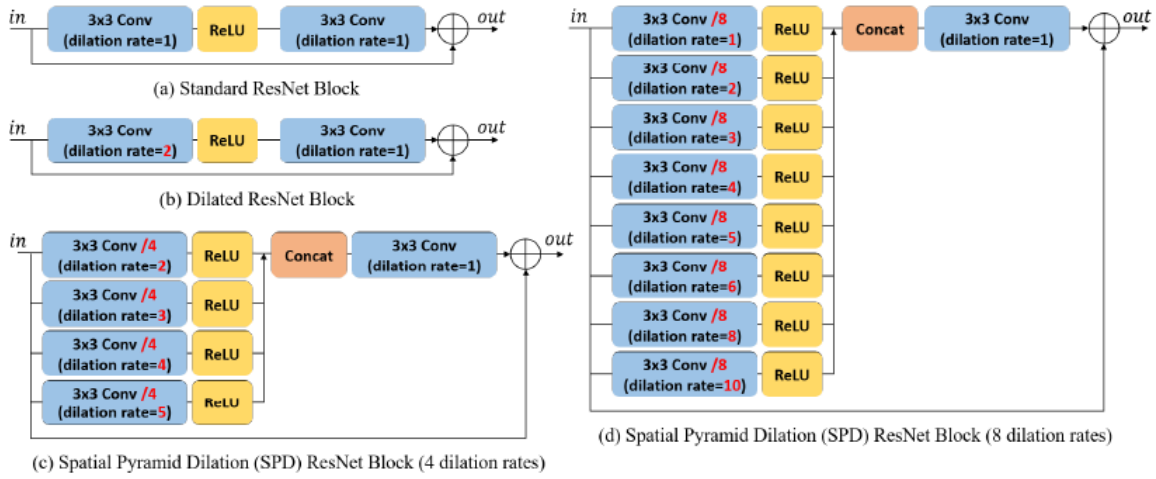


Figure 4. 2 Variations of ResNet Block. From top to bottom, left to right: (a) Standard ResNet block [84], (b) Dilated ResNet block used in [67], [68], [70] which adopts a dilation rate of 2 of the first convolutional layer, (c) The proposed SPD ResNet block with 4 dilation rates and (d) 8 dilation rates. To avoid additional parameters, we split the number of input feature channels into equal parts according to the number of dilation rates employed. As shown in (c), if 4 dilation rates are used, the output channel size of the first convolutions equals a quarter of the input channel size

Coarse Reconstruction Stage. Recall that G_1 is our coarse generator and it is responsible for rough estimation of the missing pixels in a masked image. Referring to the previous section, we concatenate $(\mathbf{I}_{in}, \mathbf{M}) \in \mathbb{R}^{H\times W\times(3+1)}$ as the input to G_1 and then obtain the coarse image $\mathbf{I}_{coarse}$. G_1 follows an encoder-decoder structure. As the scales of the masked regions are randomly determined, we proposed a Spatial Pyramid Dilation (SPD) ResNet block with various dilation rates to enlarge the receptive fields such that information given by distant spatial locations can be included for reconstruction. Our SPD ResNet block is an improved

version of the original ResNet block [84] as shown in Figure 4.2, and in total, 6 SPD ResNet blocks with 8 different dilation rates are used at the stage.

Refinement Stage. Generator G_2 is designed for refinement of $\mathbf{I}_{coarse}$ and it is similar to generator G_1. At this stage, we have 6 SPD ResNet blocks with 4 different dilation rates and a Self-Attention (SA) block in between at the middle layers. Apart from the SPD ResNet block, Multi-Scale Self-Attention (MSSA) blocks [85], [86] are used for self-similarity consideration. The SA block used in this paper is exactly the same as the one proposed in [85]. One similarity between the SA block and the contextual attention layer [62], [68] is that they both have the concept of self-similarity which is useful for amending the reconstructed patterns based on the remaining ground truth in a masked image. We apply MSSA instead of single scale SA to enhance the coherency of the completed image $\mathbf{I}_{out}$ by attending on the self-similarity of the image itself at three different scales, namely 16 × 16, 32 × 32 and 64 × 64 as shown in Figure 4.1. To avoid an excessive increase in additional parameters, we simply use standard convolutional layers to reduce the channel size before connecting to the SA blocks. The idea of Back Projection (BP) [86], [87] is also redesigned and it is used at the last decoding process of this stage (see the shaded Back Projection region in Figure 4.1). At the layer with spatial size of 64 × 64, we output a low-resolution (LR) completed image $\mathbf{I}_{lr}$ and perform BP with $\mathbf{I}_{out}$. By learning to weight the BP residual and adding it back to update $\mathbf{I}_{out}$, the generated patterns can have better alignments with the reference ground truth and hence $\mathbf{I}_{out}$ looks more coherent.

Conditional Multi-Scale Discriminators. Two discriminators D_1 and D_2 at two input scales (i.e., 256 × 256 and 128 × 128) are trained together with the generators to stimulate details of the filled regions. Combining the idea of multi-scale discriminators [57] with SN-GANs [83] and PatchGAN [56], [62], our $D_1(\mathbf{I}_{in}, \mathbf{I})$ and $D_2(\mathbf{I}_{in}, \mathbf{I})$ take the concatenation result of two RGB images as input ($\mathbf{I}$ is either $\mathbf{I}_{compltd}$ or $\mathbf{I}_{gt}$, recall that $\mathbf{I}_{compltd}$ is the same as $\mathbf{I}_{out}$ except the valid pixels are directly replaced by the ground truth) and output a set of feature maps with

size of $H/2^2 \times W/2^2 \times c$ where c represents the number of feature maps. Note that each value on these output feature maps represents a local region in the input image at two different scales. By training D_1 and D_2 to discriminate between real and fake local regions, $\mathbf{I}_{out}$ would gradually be close to its reference ground truth $\mathbf{I}_{gt}$ in terms of both appearance and semantic similarity. For achieving stable generative adversarial learning, we employ the spectral normalization layer described in [83] after each convolutional layer in D_1 and D_2.

4.1.2 Network Learning

We design our loss function based on consideration to both quantitative accuracy and visual quality of the completed images. Our loss function consists of five major terms, namely (i) a $L1$ *loss* to ensure the pixel-wise reconstruction accuracy especially if using quantitative evaluation metrics such as PSNR and mean $L1$ error to evaluate the completed images; (ii) an *adversarial loss* to urge the distribution of the completed images to be close to the distribution of the real images; (iii) the *feature perceptual loss* used in [58] that encourages each completed image and its reference ground truth image to have similar feature representations as computed by a well-trained network with good generalization like VGG-19 [41]; (iv) the *style loss* [59] to emphasize the style similarity such as textures and colours between completed images and real images; and (v) the *total variation loss* used as a regularizer in [58] to guarantee the smoothness in the completed images by penalizing its visual artifacts or discontinuities.

L1 Loss. Our $L1$ *loss* is derived from three image pairs, namely $\mathbf{I}_{coarse}$ and $\mathbf{I}_{gt}$; $\mathbf{I}_{out}$ and $\mathbf{I}_{gt}$; and $\mathbf{I}_{lr}$ and $\mathbf{I}^{lr}{}_{gt}$. Note that $\mathbf{I}^{lr}{}_{gt}$ is obtained by down-sampling $\mathbf{I}_{gt}$ by 4 times. We sum the $L1$-norm distances of these three image pairs and define our $L1$ loss, $\mathcal{L}_{L1}$, as follows:

$$\mathcal{L}_{L1} = \lambda_{hole}\mathcal{L}_{hole} + \mathcal{L}_{valid} \tag{4.1}$$

where $\mathcal{L}_{hole}$ and $\mathcal{L}_{valid}$ are the sums of the distances which are calculated only from the missing pixels and the valid pixels respectively. λ_{hole} is a weight to the pixel-wise loss within the missing regions.

Adversarial Loss. For generative adversarial learning, our discriminators are trained to rightly distinguish $\mathbf{I}_{compltd}$ from $\mathbf{I}_{gt}$ while our generators strive to cheat the discriminators of incorrect classification. We employ the hinge loss to train our model, $\mathcal{L}_{adv,G}$ and $\mathcal{L}_{adv,D}$ are computed as:

$$\mathcal{L}_{adv,G} = -\mathbb{E}_{\mathbf{I}_{in}\sim\mathbb{P}_i}\left[D_1\left(\mathbf{I}_{in}, \mathbf{I}_{compltd}\right)\right] - \mathbb{E}_{\mathbf{I}_{in}\sim\mathbb{P}_i}\left[D_2\left(\mathbf{I}_{in}, \mathbf{I}_{compltd}\right)\right] \tag{4.2}$$

$$\mathcal{L}_{adv,D} = \mathbb{E}_{\mathbf{I}_{in}\sim\mathbb{P}_i}\left[\sum_{d=1}^{2}\left[\mathrm{ReLU}\left(1 - D_d\left(\mathbf{I}_{in}, \mathbf{I}_{gt}\right)\right) + \mathrm{ReLU}\left(1 + D_d\left(\mathbf{I}_{in}, \mathbf{I}_{compltd}\right)\right)\right]\right] \tag{4.3}$$

where $\mathbb{P}_i$ represents the data distribution of $\mathbf{I}_{in}$, ReLU is the rectified linear unit defined as $f(x) = \max(0, x)$.

Perceptual Loss. Let ϕ be the well-trained loss network, VGG-19 [41], and $\phi^{\mathbf{I}}_l$ be the activation maps of the l^{th} layer of the network ϕ given an image $\mathbf{I}$. We choose five layers of the pre-trained VGG-19, namely *conv*1_1, *conv*2_1, *conv*3_1, *conv*4_1, and *conv*5_1 for computing this loss. Our $\mathcal{L}_{perceptual}$ is calculated as:

$$\mathcal{L}_{perceptual} = \sum_{l=1}^{L}\frac{\left\|\phi_l^{\mathbf{I}_{out}} - \phi_l^{\mathbf{I}_{gt}}\right\|_1}{N_{\phi_l}^{\mathbf{I}_{gt}}} + \sum_{l=1}^{L}\frac{\left\|\phi_l^{\mathbf{I}_{compltd}} - \phi_l^{\mathbf{I}_{gt}}\right\|_1}{N_{\phi_l}^{\mathbf{I}_{gt}}} \tag{4.4}$$

where $N_{\phi_l}^{\mathbf{I}_{gt}}$ indicates the number of elements in $\phi_l^{\mathbf{I}_{gt}}$ and L equals 5 as five layers are used. Here, we compute the $L1$-norm distance between the high-level feature representations of $\mathbf{I}_{out}$, $\mathbf{I}_{compltd}$ and $\mathbf{I}_{gt}$ given by the network ϕ.

Style Loss. Let $\left(\phi_l^{\mathbf{I}}\right)^{\mathrm{T}}\left(\phi_l^{\mathbf{I}}\right)$ be the Gram matrix [59] which computes the feature correlations between each activation map of the l^{th} layer of ϕ given $\mathbf{I}$, and this is also called auto-correlation matrix. We then calculate the style loss ($\mathcal{L}_{style}$) using $\mathbf{I}_{out}$, $\mathbf{I}_{compltd}$; and $\mathbf{I}_{gt}$ as:

$$\mathcal{L}_{style} = \sum_{\mathbf{I}}^{\mathbf{I}_{out},\mathbf{I}_{compltd}} \sum_{l=1}^{L} \frac{1}{C_l C_l} \left\| \frac{1}{C_l H_l W_l} \left(\left(\phi_l^{\mathbf{I}}\right)^{\mathrm{T}} \left(\phi_l^{\mathbf{I}}\right) - \left(\phi_l^{\mathbf{I}_{gt}}\right)^{\mathrm{T}} \left(\phi_l^{\mathbf{I}_{gt}}\right) \right) \right\|_1 \quad (4.5)$$

where C_l denotes the number of activation maps of the l^{th} layer of ϕ. H_l and W_l are the height and width of each activation map of the l^{th} layer of ϕ. Note that we use the same five layers of the VGG-19 as mentioned for this loss as well.

Total Variation (TV) Loss. We also adopt the total variation regularization to ensure the smoothness in $\mathbf{I}_{compltd}$.

$$\mathcal{L}_{tv} = \sum_{x,y}^{H-1,W} \frac{\left\| \mathbf{I}_{compltd}^{x+1,y} - \mathbf{I}_{compltd}^{x,y} \right\|_1}{N_{\mathbf{I}_{compltd}^{row}}} + \sum_{x,y}^{H,W-1} \frac{\left\| \mathbf{I}_{compltd}^{x,y+1} - \mathbf{I}_{compltd}^{x,y} \right\|_1}{N_{\mathbf{I}_{compltd}^{col}}} \quad (4.6)$$

where H and W are height and width of $\mathbf{I}_{compltd}$. $N^{row}{}_{\mathbf{I}_{compltd}}$ and $N^{col}{}_{\mathbf{I}_{compltd}}$ are the number of pixels in $\mathbf{I}_{compltd}$ except for the last row and the last column respectively.

Total Loss. Our total loss function for the generators is the weighted sum of the five major loss terms:

$$\mathcal{L}_{total} = \mathcal{L}_{L1} + \lambda_{adv}\mathcal{L}_{adv,G} + \lambda_{perceptual}\mathcal{L}_{perceptual} + \lambda_{style}\mathcal{L}_{style} + \lambda_{tv}\mathcal{L}_{tv} \quad (4.7)$$

where λ_{adv}, $\lambda_{pereceptual}$, λ_{style} and λ_{tv} are the hyper-parameters which indicate the significance of each term.

4.1.3 Experimental Results

We have participated in the AIM 2020 Extreme Image Inpainting Challenge [78] of the ECCV 2020 (please find in our github page for details and qualitative results of the challenge). In designing our proposed model, we take reference to the networks in [56]-[58]. We have

attached our improved SPD ResNet block to our DeepGIN. We have also modified and applied the ideas of MSSA and BP in our proposed model. Inspired by ESRGAN [55], we remove all batch normalization layers in the model to smooth out the related visual artifacts. We have used discriminators as two different scales which share the same architecture. Also, we have adjusted the number of layers of each discriminator and applied spectral normalization layers [62], [83] after the convolutional layers for training stability.

4.1.3.1 Training Procedure

Random Mask Generation. Three different types of masks are used in our training. The first type is a rectangular mask with the height and width between 30-70% of each dimension [65]-[68], [78]. The second type is the free-form mask proposed in [62]. The third type of masks is introduced in the AIM 2020 Image Inpainting Challenge [78], for which masks are randomly generated based on cellular automata. During training, each mask was randomly generated and we applied the three types of masks to each training image to get three different masked images. We observed that this can balance the three types of masks to achieve more stable training.

Training Batch Formation. As the size of training images could be every diverse, we resized all training images to the size of 512 × 512 and adopted a sub-sampling method [88] to randomly select a sub-image with size of 256 × 256. We then apply the random mask generation as stated above to obtain three masked images. Therefore, each training image becomes three training images. We set a batch size of 4 and this means that there are 12 training images in a batch.

Two-Stage Training. Our training process is divided into two stages, namely a warm-up stage and then the main stage. First, we trained only the generators by using the L1 loss for 10 epochs. We used the initialization method mentioned in [55], using a smaller initialization

for ease of training a very deep network. The trained model at the warm-up stage was used as an initialization for the main stage. This L1-oriented pre-trained model provides a reasonable initial point for training GANs, for which a balance between quantitative accuracy of the reconstruction and visual quality of the output is required. For the main stage, we trained the generators alternately with the discriminators for 100 epochs. We used Adam [89] with momentum 0.5 for both stages. The initial learning rates for generators and discriminators were set to 0.0001 and 0.0004 respectively. We trained them for 10 epochs with the initial rates and linearly decayed the rates to zero over the last 90 epochs. The hyper-parameters of the loss terms in Eqs. 4.1 and 4.7 were set to λ_{hole} = 5.0, λ_{adv} = 0.001, $\lambda_{perceptual}$ = 0.05, λ_{style} = 80.0, and λ_{tv} = 0.1. We developed our model using Pytorch 1.5.0 [90] and trained it on two NVIDIA GeForce RTX 2080Ti GPUs.

4.1.3.2 Training Data

ADE20K Dataset. We trained our model on the subset of ADE20K dataset [80], [81] for participating in the AIM challenge [78]. This dataset is collected for scene parsing and understanding, in which it contains images from various scene categories. The subset is provided by the organizers of the challenge and it consists of 10,330 training images with diverse resolutions roughly, from 256 × 256 to 3648 × 2736. We took around two and a half days for training on this dataset.

CelebA-HQ Dataset. Beyond the ADE20K dataset, we also trained our model on the CelebA-HQ dataset [82] that contains 30K high-quality face images with a standard size of 1024 × 1024. We randomly split this dataset into two groups, 27,000 images for training and 3,000 images for testing. This required approximately 6 days to train our model on this dataset.

4.1.4 Analysis of Experimental Results

We have thoroughly evaluated our proposed model. We first provide evidence in our model analysis to show the effectiveness of our suggested strategies for using Spatial Pyramid

Dilation (SPD) ResNet block, Multi-Scale Self-Attention (MSSA), and Back Projection (BP). We then compare our model with state-of-the-art approaches, namely DeepFillv1 [68] and DeepFillv2 [62], which are known to have a good generalization. We demonstrate that our model is able to handle images in the wild by testing it on two publicly available datasets, namely Flickr-Faces-HQ (FFHQ) dataset [91] and The Oxford Buildings (Oxford) dataset [92]. Related materials are available at: https://github.com/rlct1/DeepGIN.

4.1.4.1 Model Analysis

We first evaluate the effectiveness of the three proposed strategies, namely SPD, MSSA, and BP. Refer to the proposed architecture as shown in Figure 4.1, our baselines are denotated as StdResBlk (Coarse only, using only the Coarse Reconstruction Stage) and StdResBlk (a conventional ResNet of inapinting), for which all SA blocks and BP branch are eliminated and all SPD ResNet blocks are replaced by standard ResNet block (see Figure 4.2(a)). DilatedResBlk or SPDResBlk represents StdResBlk with standard ResNet blocks replaced by Dilated or SPD ResNet blocks. Please refer to Figure 4.2(b), (c) and (d). SA or MSSA indicates whether single SA block or MSSA is used and the use of BP is denoted as BP. We conducted the model analysis on CelebA-HQ dataset [82] using the 3K testing images. Note that the testing images were randomly masked by the three types of masks and the same set of masked images was used for each variation of our model. During testing, for images with size larger than 256 × 256, we divided the input into a number of 256 × 256 sub-images using the sub-sampling method [88] and obtained the completed sub-images. We then regrouped the sub-images to form the completed image by using the reverse sub-sampling method. We finally replaced the valid pixels by the ground truth.

Variations of our model	Number of parameters	PSNR	SSIM	L1 err. (%)	FID	LPIPS
StdResBlk (Coarse only)	8.168M	31.55	0.925	4.690	23.824	0.182
StdResBlk	40.850M	31.34	0.923	4.710	19.436	0.191
StdResBlk-SA	41.376M	31.60	0.925	4.510	18.239	0.180
StdResBlk-MSSA	42.892M	32.66	0.933	4.067	12.843	0.148
DilatedResBlk-MSSA	42.892M	32.71	0.933	4.034	12.548	0.149
SPDResBlk-MSSA	42.892M	32.88	0.935	3.884	12.335	0.143
SPDResBlk-MSSA-BP	42.930M	**33.26**	**0.939**	**3.666**	**11.424**	**0.132**

Table 4. 1 Model analysis of our proposed model on CelebA-HQ dataset. The best results are in **bold** typeface

Quantitative Comparisons. As the lack of good quantitative evaluation metric for inpainting [61], [62], [68], we report several numerical metrics which are commonly used in image manipulation, namely PSNR, SSIM [93], mean L1 error, Fréchet Inception Distance (FID) [94], and Learned Perceptual Image Patch Similarity (LPIPS) [95], for a comprehensive analysis of the performance. The results are listed in Table 4.1 and higher PSNR, SSIM and smaller L1 error mean better pixel-wise reconstruction accuracy. FID and LPIPS are also used to estimate the visual quality of the output, the smaller the better. It is obvious that our full model, SPDResBlk-MSSA-BP, gives the best performance on these numerical metrics. The employment of MSSA brings an 1.06 dB increase in PSNR compared to StdResBlk-SA. This reflects the importance of multi-scale self-similarity to inpainting. Our SPD ResNet blocks and the adoption of BP also bring about 0.22 dB and 0.38 dB improvement in PSNR respectively.

Figure 4. 3 Comparisons of test results of the variations of our model on CelebA-HQ dataset. Three different types of masked images are displayed from top to bottom. The first and the last columns show $\mathbf{I}_{in}$ and $\mathbf{I}_{gt}$ respectively. The variations of our model are indicated on top of the figure. Our full model (the 8th column), SPDResBlk-MSSA-BP (GAN-based), provides high quality results with both the best similarity and visual quality to the ground truth images. Please zoom in for a better view

Qualitative Comparisons. Figure 4.3 shows the comparisons of the variations of our model on CelebA-HQ dataset. Without the second refinement stage, i.e., Coarse only, the completed images lack for facial details like the first example of the 2nd column in Figure 4.3. We can see the blurriness of the completed face. It can also be observed that the use of MSSA greatly enhances the visual quality as compared to the two which are without SA block and with only a single SA block (see the 3rd and 4th columns). Apart from this, with the SPD ResNet blocks and BP technique, the completed images are with better colour coherency and alignment of the generated features. For example, see the spectacle frames and the eyes in the 2nd and 3rd rows respectively.

4.1.4.2 Comparison with Previous Works

In order to test the generalization of our model, we compare our best model against some state-of-the-art approaches, DeepFillv1 [68] and DeepFillv2 [62], on the two publicly available datasets, FFHQ [91] and Oxford Buildings [92]. It is worth nothing that both DeepFillv1 and v2 are known to have good generalization for dealing with images in the wild.

We directly used their provided pre-trained models[2] for comparison. The FFHQ dataset is similar to the CelebA-HQ dataset and it contains 70K high-quality face images at 1024 × 1024 resolution. We randomly selected 1,000 images for the testing on this dataset. For the Oxford dataset, it consists of 5,062 images of Oxford landmarks with a wide variety of styles. The images include buildings, suburban areas, halls, people, etc. we also randomly selected 523 testing images on this dataset for comparison.

Similarly, testing images were randomly masked by the three types of masks. For DeepFillv1 and v2, the authors divided an image into a number of grids to perform inpainting and indicated that their models were trained with images of resolution 256 × 256. Note also that DeepFillv1 was trained only for the rectangular types of masks. For fair comparison, we also conducted experiments in which testing images were randomly masked only by the rectangular masks.

[2] https://github.com/JiahuiYu/generative_inpainting

Method	PSNR	SSIM	L1 err. (%)	FID	LPIPS
Flickr-Faces-HQ-Dataset (FFHQ), random rectangular masks					
DeepFillv1 (OS)	20.22	0.872	16.523	97.630	0.173
DeepFillv2 (OS)	20.95	0.903	14.607	92.070	0.170
Ours (OS)	**26.05**	**0.923**	**7.183**	**20.849**	**0.137**
DeepFillv1 (256)	21.55	0.836	13.631	26.276	0.144
DeepFillv2 (256)	22.52	0.845	12.029	**19.336**	**0.128**
Ours (256)	**24.36**	**0.867**	**9.797**	37.577	0.142
The Oxford Buildings Dataset (Oxford), random rectangular masks					
DeepFillv1 (OS)	19.20	0.767	20.322	67.193	0.187
DeepFillv2 (OS)	18.58	0.766	21.204	77.636	0.192
Ours (OS)	**21.92**	**0.861**	**12.067**	**63.744**	**0.170**
DeepFillv1 (256)	19.49	0.795	16.851	**58.588**	**0.169**
DeepFillv2 (256)	18.88	0.789	18.308	66.615	0.174
Ours (256)	**21.90**	**0.819**	**12.995**	74.866	0.185
Flickr-Faces-HQ-Dataset (FFHQ), random three types of masks					
DeepFillv1 (OS)	25.12	0.839	11.363	64.534	0.232
DeepFillv2 (OS)	29.70	0.912	7.994	36.940	0.188
Ours (OS)	**32.36**	**0.929**	**4.071**	**14.327**	**0.156**
DeepFillv1 (256)	22.87	0.683	16.812	80.952	0.310
DeepFillv2 (256)	22.75	0.716	17.472	75.555	0.293
Ours (256)	**24.71**	**0.760**	**13.417**	**64.542**	**0.274**
The Oxford Buildings Dataset (Oxford), random three types of masks					
DeepFillv1 (OS)	21.48	0.741	23.460	61.958	0.237
DeepFillv2 (OS)	24.68	0.802	19.195	38.315	**0.179**
Ours (OS)	**27.57**	**0.871**	**7.268**	**38.016**	0.191
DeepFillv1 (256)	21.64	0.686	18.835	81.009	0.284
DeepFillv2 (256)	20.80	0.702	20.687	82.671	0.266
Ours (256)	**23.60**	**0.744**	**14.659**	**79.927**	**0.265**

Table 4. 2 Comparisons of DeepFillv1 [68], and DeepFillv2 [62] on both FFHQ and Oxford datasets with two sets of masked images. One set only contains the rectangular masks while another set includes all the three types of masks. Our DeepGIN is denoted as Ours (i.e., the full model, SPDResBlk-MSSA-BP in the previous section). (OS) and (256) mean that the testing images are with the original sizes and size of 256 × 256 respectively. The best results are in **bold** typeface

Quantitative Comparisons. Table 4.2 shows the comparisons with DeepFillv1 and DeepFillv2 on the two datasets with two sets of masked images. It is clear that our model outperforms DeepFillv1 and v2 in all the experiments on the two datasets in terms of the pixel-wise reconstruction accuracy. Our model achieves better PSNR compared with DeepFillv1 and v2 in the range of 1.84~5.1 dB and offers better SSIM and L1 error. For FID and LPIPS, we attain better performance on the testing images with the original sizes. For the testing images with size of 256 × 256 and masked by random rectangular masks, we are also comparable to the other two approaches.

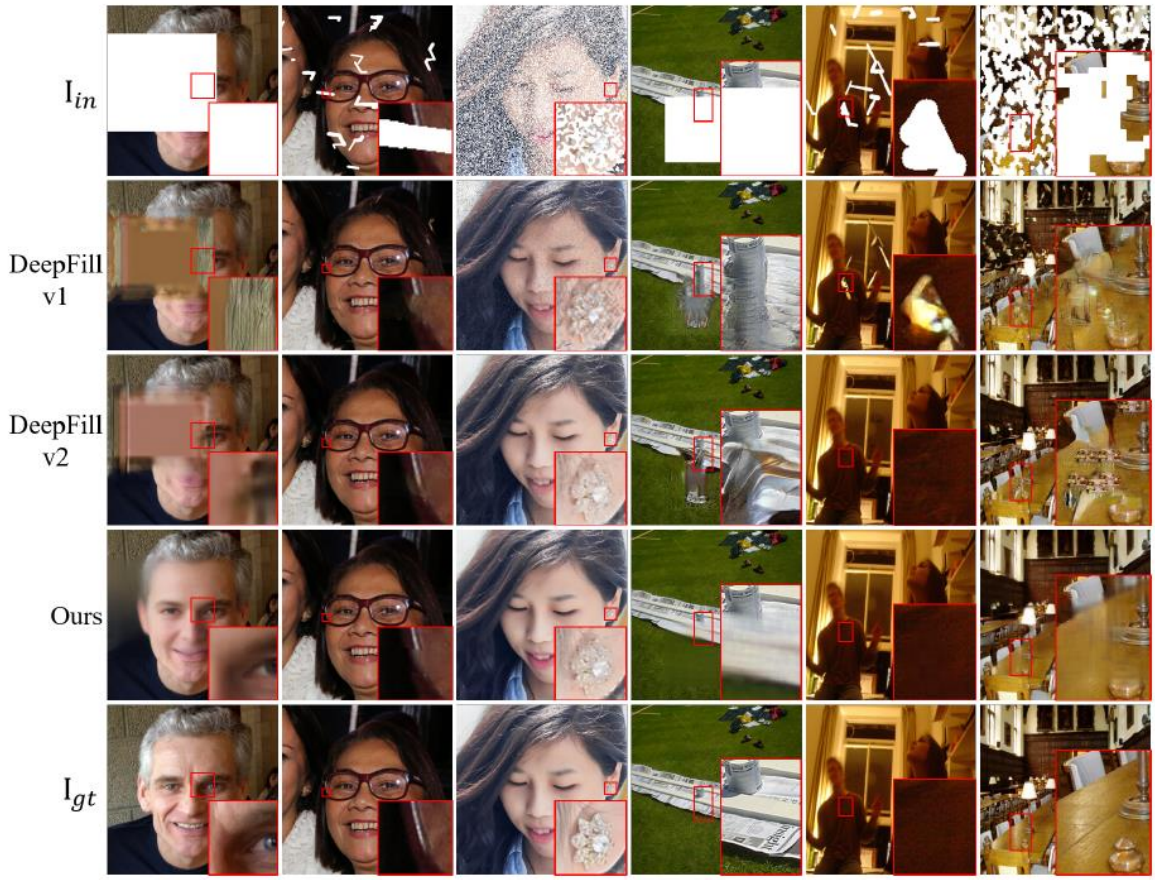


Figure 4. 4 Comparisons of test results on FFHQ and Oxford Buildings datasets. Each column shows an example of the test results. From top to bottom: the first row displays various masked input images ($\mathbf{I}_{in}$) from both datasets. The second to the fourth rows show the completed images by DeepFillv1, v2 and our DeepGIN respectively. The reference ground truth images ($\mathbf{I}_{gt}$) are also provided at the last row. Please zoom in for a better view

Qualitative Comparisons. Figure 4.4 displays the test results on both FFHQ and Oxford datasets. It can be seen that DeepFillv1 and DeepFillv2 fail to achieve satisfactory visual quality on the large rectangular masks as shown in the first and fourth columns in Figure 4.4. For the other two types of masked images, our model is also able to provide the completed images with better colour and content coherency. Note that our model tends to produce blurry images and the reason is that our model was trained to be more PSNR-oriented than the DeepFillv1 and DeepFillv2. We seek a balance between the pixel-wise accuracy and the visual quality to avoid some strange generated patterns like the completed table image by DeepFillv2 (3rd row) of the last example (last column) in Figure 4.4.

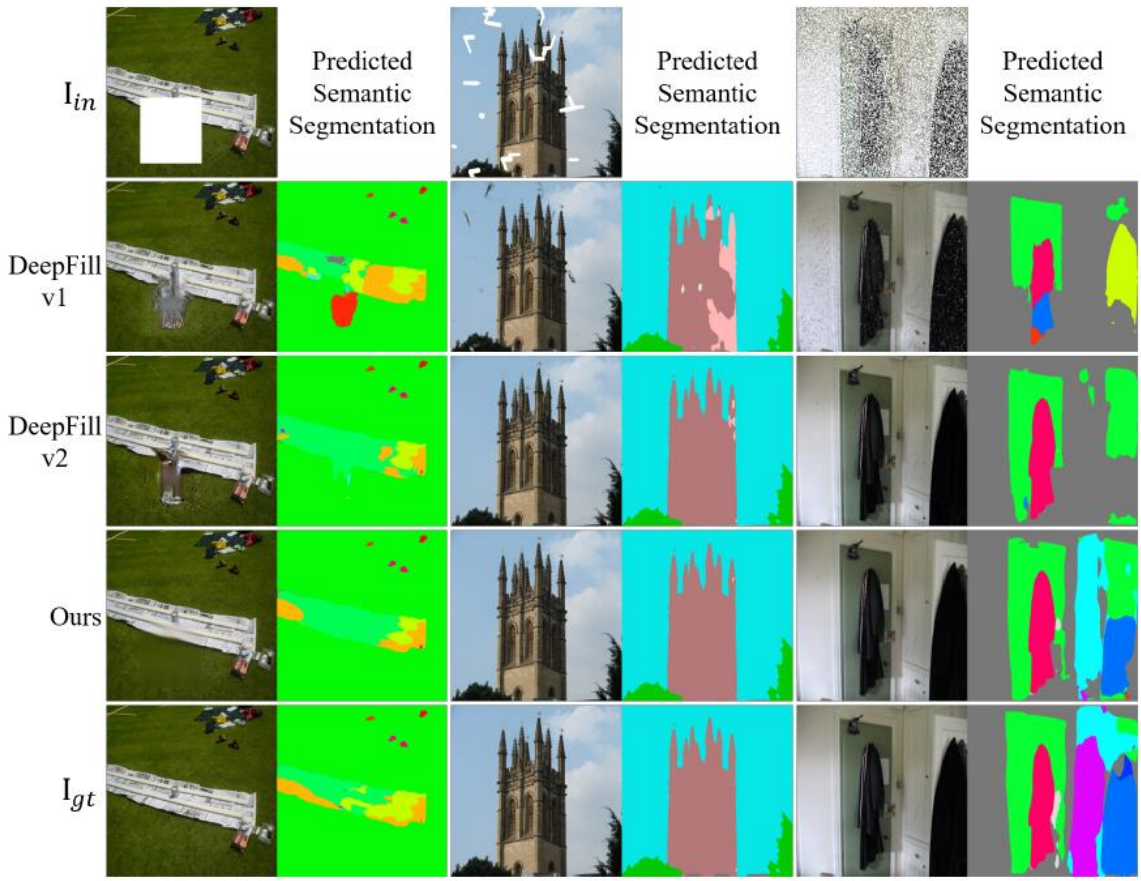


Figure 4. 5 Visualizations of predicted semantic segmentation test results on Oxford Buildings dataset. The 2nd to 4th rows show the completed images by different methods and the corresponding predicted semantic segmentation obtained using the trained semantic segmentation network [81]. The ground truth and segmentation images are also attached to the last row for readers' reference. Please zoom in for a better view of the results

To show that our model offers better pixel-wise reconstruction accuracy and our completed images benefit other high-level computer vision tasks, we provide the predicted semantic segmentation test results as shown in Figure 4.5. It is obvious that our results are semantically closer to $\mathbf{I}_{gt}$ (the last row) than that of the other two methods, i.e., DeepFillv1 (2nd row) and DeepFillv2 (3rd row), see for example, the intersection of the newspaper and the lawn in Figure 4.5. Semantic correctness of the completed images is important when the completed images would be further used for other high-level tasks. For example, the completed images with dynamic and/or static object removal can be used for place recognition tasks.

4.1.5 Conclusion on Generative Inpainting

We have presented a deep generative inpainting network, called DeepGIN. Unlike the existing works, we propose a Spatial Pyramid Dilation (SPD) ResNet block to include more receptive fields for utilizing information given by distant spatial locations. This is important to inpainting especially when the masked regions are too large to be filled. We also enhance the significance of self-similarity consideration, hence we employ Multi-Scale Self-Attention (MSSA) strategy to enhance our performance. Furthermore, Back Projection (BP) is strategically used to improve the alignment of the generated and valid pixels. We have achieved performance better than the state-of-the-art image inpainting. This research work participated in the AIM 2020 Extreme Image Inpainting Challenge, which requires the right balance of pixel-wise reconstruction accuracy and visual quality. We believe that our DeepGIN is able to achieve the right balance and we encourage scholars in the field to give more attention in this direction.

4.2 General Image Inpainting with Advanced Global Semantics and Visual Quality

Let us firstly define an input RGB masked image and a binary mask image as $\mathbf{I}_{in} \in \mathbb{R}^{3\times H\times W}$ and $\mathbf{M} \in \mathbb{R}^{1\times H\times W}$ respectively. Pixels with value 1 in $\mathbf{M}$ indicate the missing regions. $\mathbf{I}_c \in \mathbb{R}^{3\times H\times W}$ is the output of our generator network G. We also define $\mathbf{I}_{out} \in \mathbb{R}^{3\times H\times W}$ which is the same as $\mathbf{I}_c$ except that pixels located in the known regions outside the mask are directly replaced by the original valid pixels. $\mathbf{I}_{gt} \in \mathbb{R}^{3\times H\times W}$ is the ground truth image. Our objective is to fill in $\mathbf{I}_{in}$ with its condition $\mathbf{M}$ and the output $\mathbf{I}_{out}$ should be both semantically and visually similar to $\mathbf{I}_{gt}$. Our proposed model is designed to be trained under the framework of GANs with training data $\{\mathbf{I}_{in}, \mathbf{M}, \mathbf{I}_{gt}\}$. Note that the training mask $\mathbf{M}$ is randomly generated with arbitrary types and sizes. Our GAN generator G takes $\mathbf{I}_{in}$ and $\mathbf{M}$ as input and generate $\mathbf{I}_c$ as output. We then produce

$\mathbf{I}_{out} = \mathbf{I}_c \odot \mathbf{M} + \mathbf{I}_{in} \odot (\mathbf{1}-\mathbf{M})$ as the ultimate output, where $\odot$ represents elementwise, i.e. element by element, multiplication.

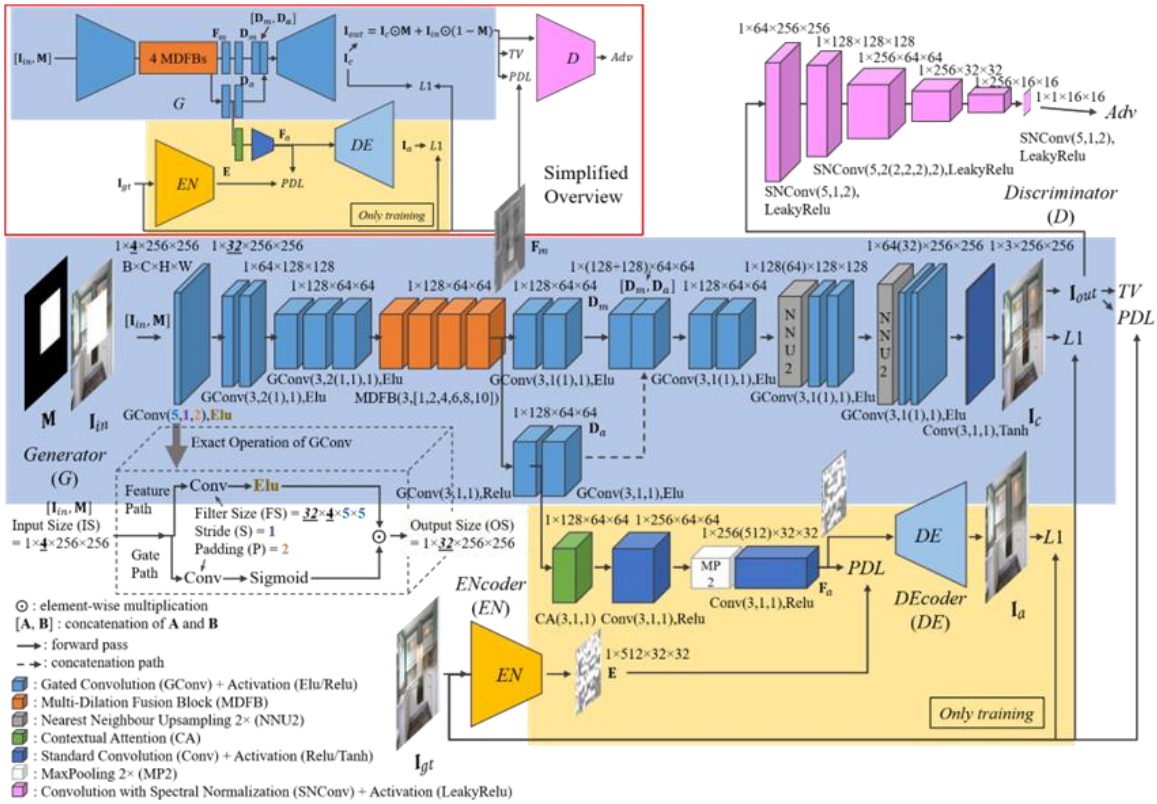


Figure 4. 6 An overview of the proposed model for image inpainting. Refer to the top-left red-bounded simplified overview diagram, our model contains a *G*enerator *G* (light blue shaded region), a well pre-trained autoencoder which is decomposed into an *EN*coder (*EN*, yellow trapezoid) and a *DE*coder (*DE*, light blue trapezoid), and a standard SN-PatchGAN *D*iscriminator (*D*, pink trapezoid). Only the generator (*G*) is used in both training and testing. The dashed block next to *G*enerator (*G*) shows the exact operation of a gated convolution. Also, the auxiliary contextual attention learning branch (cream-shaded area) is only activated during training. This means that this branch is not used for testing.

An overview of our proposed model is shown in Figure 4.6. The top-left red-bounded corner of Figure 4.6 gives a simplified overview of our model. For training, three networks are involved, namely *G*enerator *G* (the whole light blue shaded area), a well pre-trained autoencoder which consists of an *EN*coder (*EN*, yellow trapezoid) and a *DE*coder (*DE*, light blue trapezoid), and a standard SN-PatchGAN *D*iscriminator (*D*, pink trapezoid). For inference, only the generator network (*G*) is used. It is responsible for producing realistic inpainting results. The objective of *D* is to identify real images from images generated by *G* during

training for adversarial learning. Generally speaking, *G* has to learn to produce realistic images to fool *D*, while *D* has to learn to discriminate between real and generated images. Under this adversarial framework, *G* would eventually be able to generate realistic images. *EN* and *DE* are the encoder and decoder of a well pre-trained autoencoder network, say introduced in [58] for the tasks of style transfer and super-resolution. We suggest to use them as auxiliary networks for obtaining the target encoded features (**E**) to optimize a perceptual loss (*PDL*) and decoding our learned decoded features ($\mathbf{F}_a$) to get decoded images ($\mathbf{I}_a$) to minimize the per-pixel ($L1$) loss in our auxiliary contextual attention learning branch (cream-shaded areas in Figure 4.6). Note that the parameters of the pre-trained *EN* and *DE* are fixed during our training.

4.2.1 Network Design

4.2.1.1 Generator Network

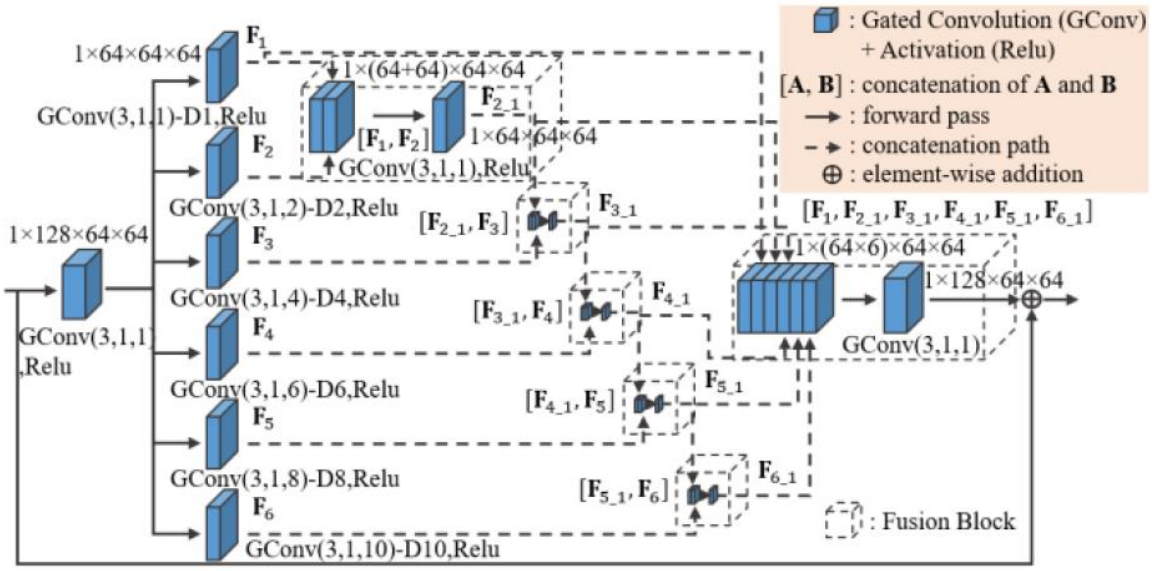


Figure 4. 7 The structure of the proposed Multi-Dilation Fusion Block (MDFB). Residual block and gated convolutions with multiple dilation rates are the basic components of our MDFB.

The generator (*G*, light blue shaded area) essentially has a standard encoder-decoder network structure. It takes masked input image $\mathbf{I}_{in}$ and its binary mask image **M** as input. The missing pixels in $\mathbf{I}_{in}$ are indicated by value 0. Input to *G* is encoded to features ($\mathbf{F}_m$) with channel size of 128 and spatial size of 64×64 (i.e. 4× downsampling the input). Encoded features ($\mathbf{F}_m$)

are fed to the main branch and the CA branch to obtain decoded features $\mathbf{D}_m$ and $\mathbf{D}_a$ respectively. $\mathbf{D}_m$ and $\mathbf{D}_a$ are then concatenated and decoded to obtain the completed image $\mathbf{I}_c$ and the output $\mathbf{I}_{out}$. As gated and dilated convolutions have been proven to be useful for indicating the validity of each spatial location and understanding the context of an image respectively, we propose a Multi-Dilation Fusion Block (MDFB) to be used as a major processing unit in the proposed *G*, i.e. the 4 orange blocks as shown in the middle of the light blue shaded area of Figure 4.6. The structure of MDFB is shown in Figure 4.7 and it is a dedicated new design with residual connections [84]. In Figure 4.7, six parallel gated convolutions (their results are denoted as $\mathbf{F}_1$ to $\mathbf{F}_6$) with different dilation rates (i.e. d = 1, 2, 3, 4, 6, 8 and 10) are used to enlarge the receptive fields such that information from both near and distant spatial locations can be captured for reconstructing local missing regions. The conventional approaches use consecutive single dilated convolutional layers [62], [68], [76] or multiple dilated convolutional layers followed by a single aggressive concatenation layer [75]. Different from those, we intentionally fuse the convolutional results from each pair of consecutive dilation rates (the five dashed fusion blocks in the middle of Figure 4.7) to capture the neighboring information according to the closeness of the spatial locations, from near to distant spatial locations. We obtain five fused features $\mathbf{F}_{2\_1}$, $\mathbf{F}_{3\_1}$, $\mathbf{F}_{4\_1}$, $\mathbf{F}_{5\_1}$ and $\mathbf{F}_{6\_1}$ from the five dashed fusion blocks. We then concatenate the fused features with multiple scales and perform convolution to get the residues as shown in the 6th rightmost dashed fusion block of Figure 4.7. Finally, the residues are added to the input via the residual connection. This is the output of our MDFB. We use 4 MDFBs to obtain encoded features $\mathbf{F}_m$ (i.e. the output coming out from the 4th MDFB orange block at the middle of the light blue shaded area in Figure 4.6). In so doing, the generated features inside the missing regions would have better global semantics.

Contextual Attention (CA) (Green block in the cream-shaded region of Figure 4.6): In order to fully utilize the information given by the known regions, we design an auxiliary

contextual attention (CA) learning branch as shown in the cream-shaded region of Figure 4.6. Actually, the CA operation is similar to the non-local operation (also known as self-attention) introduced in [85]. Note that the self-attention is a point-wise operation while CA is a patch-wise operation. More specifically, we extract reference feature patches from the known regions to form a number of filters (kernels) and then convolve with the generated feature patches (patches located at the missing regions) through standard convolutional operation. Then, we obtain the convolutional results with channel size same as the number of extracted filters and each channel represents the correlations between all the generated feature patches and the corresponding extracted reference feature patch (filter). We perform softmax operation along the channel dimension and get the similarities between each generated feature patch and all the reference feature patches. We rebuild the generated feature patches by weighted sums of the reference feature patches and the weights are derived from similarities. The similarities range from 0 to 1 and higher similarity means higher weight. This means that each generated feature patch is replaced by a weighted combination of all the extracted reference feature patches. The output of the CA operation is with the same size as the input, i.e. 1×128×64×64 as shown in the green block of Figure 4.6.

From the bottom cream-shaded area in Figure 4.6, after the CA operation, standard convolutional layers and a max pooling layer are applied to obtain our learned decoded features ($\mathbf{F}_a$) with size of 1×512×32×32. We intentionally encourage $\mathbf{F}_a$ to be close to the target encoded features (**E**) as computed by the encoder (*EN*) of the well pre-trained autoencoder [58]. We compare $\mathbf{F}_a$ with **E** using a recent projected feature distribution loss (*PDL*). We also feed $\mathbf{F}_a$ to the corresponding decoder (*DE*) of the well pre-trained autoencoder to obtain the decoded images $\mathbf{I}_a$. If $\mathbf{F}_a$ have similar feature representations to **E**, $\mathbf{I}_a$ should be close to $\mathbf{I}_{gt}$. Hence, we also compute a per-pixel loss (*L1*) between $\mathbf{I}_a$ and $\mathbf{I}_{gt}$. By so doing, this CA branch has to learn to use the known feature patches to reconstruct the generated feature patches. Let us introduce

the function of the well pre-trained autoencoder and our reason for using it in the following sub-section.

4.2.1.2 Autoencoder Network

We suggest to use a well pre-trained autoencoder [58] to help us learning better decoded features ($\mathbf{F}_a$). We decompose it into an encoder (*EN*) and a decoder (*DE*) as shown in the bottom yellow trapezoid and light blue trapezoid of Figure 4.6 respectively. To enhance the semantics and textures of the inpainting results, some previous inpainting models have adopted perceptual losses to force their completed images to have similar feature representations to the real images computed by typical well pre-trained CNN models such as VGG16 [41] and AlexNet [35]. We suggest to use the well pre-trained VGG16-like autoencoder for computing our perceptual loss (*PDL*, the bottom cream-shaded region of Figure 4.6). In [58], the VGG16 was modified to an autoencoder with symmetric encoder and decoder networks to reconstruct the input. Its encoded features at the bottleneck layer have to contain all the information of the input so as to perfectly reconstruct the input. Therefore, we encourage our learned features ($\mathbf{F}_a$) at the auxiliary CA learning branch to have similar projected distribution [79] as the target encoded features computed by the encoder (*EN*) of the autoencoder. In addition, if we feed $\mathbf{F}_a$ to the corresponding decoder (*DE*) of the autoencoder, we should also obtain the input real images. With this assumption, the auxiliary CA branch has to learn using the feature patches from the known regions to fill in the missing regions such that the completed images look close to the real images. This information is useful for improving the quality of our ultimate inpainting results.

Figure 4.8 shows a sample visualization of the input, encoded features at the bottleneck of our generator ($\mathbf{F}_m$), learned decoded features at the output of the auxiliary CA branch ($\mathbf{F}_a$) and the corresponding decoded image ($\mathbf{I}_a$) using *DE*, and also the target encoded features ($\mathbf{E}$) as well as the corresponding decoded image. Note that $\mathbf{E}$ is obtained by feeding $\mathbf{I}_{gt}$ to *EN*. The

locations of all these features and images are marked in Figure 4.6 and they are mainly located at the bottom cream-shaded region of Figure 4.6. Refer to Figure 4.8, for an input masked image $\mathbf{I}_{in}$ (first row first column), we look for a completed image $\mathbf{I}_{out}$ (second row first column) which should be close to the ground truth $\mathbf{I}_{gt}$ (second row second column). With the use of our proposed MDFB, we can see that the middle generated content of the encoded features $\mathbf{F}_{m}$ (first row second column, the output of the 4th MDFB orange block in Figure 4.6) are with good global semantics. Note that good alignment of the cabinet and the table can be observed. In addition, our learned decoded features $\mathbf{F}_{a}$ (first row third column) is close to **E** (second row third column). We can also see that the decoded image $\mathbf{I}_{a}$ (first row last column) is already a plausible prediction of the ground truth $\mathbf{I}_{gt}$ (second row second column). This means that our auxiliary CA learning branch is able to reconstruct the real images by using the CA operation. Note that the decoded image using **E** as encoded features is also provided (second row last column) for reference.

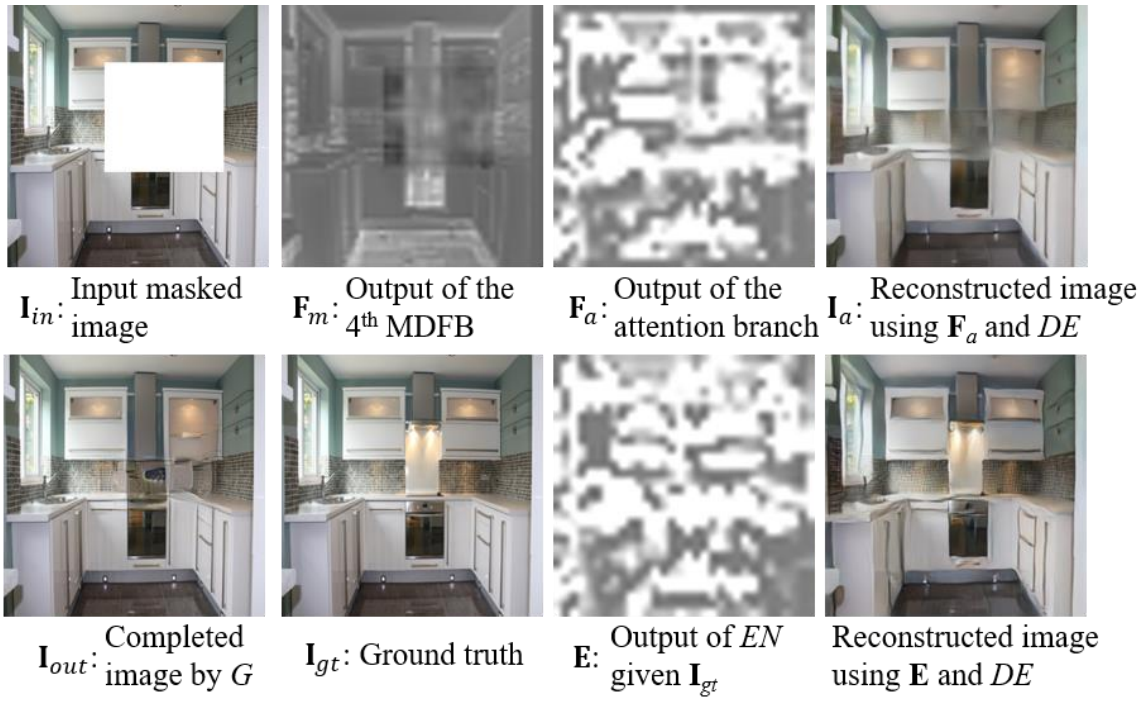


Figure 4. 8 Visualizations of input, learned features, target encoded features and output of our proposed model. $\mathbf{I}_{in}$, $\mathbf{F}_m$, and $\mathbf{I}_{out}$ are located at the middle light blue shaded area in Figure 4.6. $\mathbf{F}_a$, $\mathbf{I}_a$, and $\mathbf{E}$ are located at the bottom cream-shaded area in Figure 4.6, and $\mathbf{I}_{gt}$ is located at the bottom in Figure 4.6

4.2.1.3 Discriminator Network

We employ a SN-PatchGAN *D*iscriminator (*D*) [62], [70], [71], [74]-[76] in our training to stimulate realistic details of the generated regions. The structure of *D* is shown in the top-right corner (pink convolutional blocks) of Figure 4.6. Spectral Normalization (SN) [83] is applied to each convolutional layer of the PatchGAN discriminator [56], [57] for stable generative adversarial learning. Our *D* takes $\mathbf{I}_{out}$ or $\mathbf{I}_{gt}$ as input and outputs a single feature map with size of 16×16. Each element on this 16×16 feature map represents a local region of the input as compared to approaches in [65], [67] using discriminators which examine the entire input as a whole. By training *D* to discriminate different local patches between $\mathbf{I}_{out}$ and $\mathbf{I}_{gt}$, $\mathbf{I}_{out}$ would eventually look realistic with better local details as compared to that of examining the entire input as a whole.

4.2.2. Network Learning

The design of our loss function is based on consideration to both per-pixel reconstruction accuracy and visual quality of the inpainting results. Our total loss contains four terms, namely (i) $L1$ loss for per-pixel reconstruction accuracy; (ii) *Adversarial* (*GAN*) loss for similar feature distribution between the completed images and the real images; (iii) *Projected Distribution* loss (*PDL*) introduced in [79] that minimizes the 1D Wasserstein distance between each feature map of the completed images and the real images as computed by well pre-trained CNN models; and (iv) *Total Variation* (*TV*) loss employed in [58] as a regularization term to encourage spatial smoothness in the inpainting results. Hence obvious visual artifacts and discontinuities can be alleviated.

***L*1 loss.** Our $L1$ loss is derived from two image pairs, the exact output of our generator $\mathbf{I}_c$ and ground truth $\mathbf{I}_{gt}$; and decoded image $\mathbf{I}_a$ at the auxiliary contextual attention learning branch and $\mathbf{I}_{gt}$. We sum the $L1$ norm distances of these two image pairs and define the $L1$ loss term, L_{L1}, as follows.

$$L_{L1} = \lambda_{hole} L_{hole} + \lambda_{valid} L_{valid} \tag{4.8}$$

where L_{hole} and L_{valid} are the sums of the absolute differences calculated from the missing and the known (valid) regions respectively. λ_{hole} = 1.0 and λ_{valid} = 1.2 are the weights to the per-pixel loss for the missing and known regions respectively.

***Adversarial* loss.** We use a standard hinge loss to train our proposed model following the previous inpainting models [62], [74]-[76]. Compared to the typical logarithmic loss which maximizes Binomial likelihood for a two-class classification, hinge loss can focus on those hard samples (images which are difficult to be discriminated) by neglecting the samples which are far away from the decision boundary. This means that better accuracy is achieved by

releasing the aim of getting better probability estimation. We define the adversarial loss terms for our generator and discriminator, $L_{Adv,G}$ and $L_{Adv,D}$, as below.

$$L_{Adv,G} = -\mathbb{E}_{\mathbf{I}_{in}\sim\mathbb{P}_i}[D(\mathbf{I}_{out})] \tag{4.9}$$

$$L_{Adv,D} = \mathbb{E}_{\mathbf{I}_{gt}\sim\mathbb{P}_g}\left[\mathrm{ReLU}\left(\mathbf{1} - D\left(\mathbf{I}_{gt}\right)\right)\right] + \mathbb{E}_{\mathbf{I}_{in}\sim\mathbb{P}_i}\left[\mathrm{ReLU}\left(\mathbf{1} + D(\mathbf{I}_{out})\right)\right] \tag{4.10}$$

where $\mathbb{P}_i$ and $\mathbb{P}_g$ represent the data distributions of $\mathbf{I}_{in}$ and $\mathbf{I}_{gt}$ respectively. $D(\mathbf{I})$ means a forward pass of $\mathbf{I}$ to D and ReLU is the standard rectified linear unit function defined as $f(x) = \max(0, x)$.

***Projected Distribution* loss.** Given two non-negative vectorized feature maps $\mathbf{a}$ and $\mathbf{b} \in \mathbb{R}^{1\times HW}$ where H and W are the height and width of the two feature maps. We can compute their cumulative sums $\mathbf{A}$ and $\mathbf{B}$ such that $A_i \leq A_{i+1}$ and $B_i \leq B_{i+1}$, where $i = [0, HW-1]$. Then, the 1D Wasserstein distance between two projected one-dimensional feature maps can be computed as.

$$W_1(\mathbf{A}, \mathbf{B}) = \sum_{i=0}^{HW-1}|A_i - B_i| \tag{4.11}$$

The projected distribution loss is to calculate the sum of the absolute difference between the cumulative sums of two non-negative vectorized feature maps. Note that non-negative feature maps can be extracted from most CNN models after any ReLU activation layer. For a concise notation, we simply use $W_1(\mathbf{a},\mathbf{b})$ to represent the 1D Wasserstein distance between two non-negative feature maps. Note that it also includes the vectorization and cumulative sum operation. Compared to directly computing the mean difference ($L1$) between feature maps, the cumulative sum operation can better capture the geometric information on features. For a cumulative sum of a non-negative vectorized feature map, the locations where have high activation values are highlighted by obvious increase in the cumulative sum. To minimize the absolute difference between two cumulative sums, the locations with high activation values

should be the same. Hence, geometrics of features are captured to obtain better learned features as compared to $L1$ which merely aims for minimizing the mean difference between feature maps.

Let φ be a well pre-trained CNN model and $\varphi(\mathbf{I})$ be the features of a particular layer of the model φ when the image $\mathbf{I}$ is forward passed to φ. For the features with size $C \times H \times W$, where C, H, and W are the channel size, height and width of the features, we define our projected distribution losses for the auxiliary contextual attention learning branch and the main branch as $L_{a,PD}$ and $L_{m,PD}$, and they are computed as:

$$L_{a,PD} = \sum_{i=0}^{C} W_1\left(\mathbf{F}_a^i, \varphi^i\left(\mathbf{I}_{gt}\right)\right) \tag{4.12}$$

$$L_{m,PD} = \sum_{i=0}^{C} W_1\left(\varphi^i(\mathbf{I}_{out}), \varphi^i\left(\mathbf{I}_{gt}\right)\right) \tag{4.13}$$

$$L_{PD} = \lambda_{a,PD} L_{a,PD} + \lambda_{m,PD} L_{m,PD} \tag{4.14}$$

where $\mathbf{F}^i_a$ is the i^{th} feature map out of the entire set of our learned features ($\mathbf{F}_a$, the bottom cream-shaded area of Figure 4.6) and $\varphi^i(\mathbf{I})$ is the i^{th} feature map out of the entire set of feature maps. Note that φ is the encoder (*EN*) of the well pre-trained autoencoder [58] and we use its bottleneck features for computing $L_{a,PD}$. For $L_{m,PD}$, φ is the well pre-trained VGG16 [41] and we suggest to use the relu4_2 features for computing $L_{m,PD}$. Note $\lambda_{a,PD} = 0.01$ and $\lambda_{m,PD} = 0.005$ are good selected weights for $L_{a,PD}$ and $L_{m,PD}$ respectively.

***Total Variation* loss.** We also use the total variation (*TV*) loss, L_{TV}, to ensure the smoothness in $\mathbf{I}_{out}$.

$$L_{TV} = \sum_{x,y}^{H-1,W} \frac{\left|\mathbf{I}_{out}^{x+1,y} - \mathbf{I}_{out}^{x,y}\right|}{N_{\mathbf{I}_{out}}^{row}} + \sum_{x,y}^{H,W-1} \frac{\left|\mathbf{I}_{out}^{x,y+1} - \mathbf{I}_{out}^{x,y}\right|}{N_{\mathbf{I}_{out}}^{col}} \tag{4.15}$$

where H and W are the height and width of $\mathbf{I}_{out}$. The terms $N^{row}{}_{\mathbf{I}_{out}}$ and $N^{col}{}_{\mathbf{I}_{out}}$ are the number of pixels in $\mathbf{I}_{out}$ except for the last row and the last column respectively.

Total loss. The total loss function, L_{Total}, for training our model is the weighted sum of the four major loss terms:

$$L_{Total} = L_{L1} + L_{PD} + \lambda_{Adv} L_{Adv,G} + \lambda_{TV} L_{TV} \quad (4.16)$$

where λ_{Adv} and λ_{TV} are the weights to indicate the significance of the *adversarial* and *TV* loss terms respectively.

4.2.3 Experimental Details

In this section, we first describe our training procedure which includes how we generate random masks, how we form training mini-batches and how we train our model. Then, we present our training data to train our models for face and general image inpainting.

4.2.3.1 Training Procedure

Arbitrary masks. Both regular and irregular masks were included in our training. For regular masks, rectangular masks with the height and width ranging from 30% to 70% of the image size were used following previous methods [62], [65]-[76]. For irregular masks, we used the free-form masks proposed in [62]. We also included the third type of masks which was introduced in the AIM 2020 extreme image inpainting challenge [78]. This type of masks is arbitrarily generated based on cellular automata. During training, masks were randomly generated and applied to each training image to obtain three different masked images.

Training mini-batch. All the training images were resized to 256×256. We then applied the random masking to each training image as stated above. Therefore, each training image became three training images, one was regularly masked, another one was irregularly masked and the last one was masked based on cellular automata. We set the mini-batch size to 4, hence 12 (= 4×3) training images formed a mini-batch.

Training details. We developed our proposed model using PyTorch 1.5.0 [90] and trained the model on two NVIDIA GeForce RTX 2080Ti GPUs. We used a smaller

initialization for ease of training a deep network following [55]. We firstly trained only the generator network using the $L1$ loss for 5 epochs as it could act as a reasonable initialization for the following adversarial training. Then, we trained the generator alternatively with the discriminator for 95 epochs. We used Adam [89] with momentum 0.5 and the initial learning rate was set to 0.0001. The learning rate was halved starting at the 20^{th}, 40^{th}, 60^{th}, and 80^{th} epochs. The weights of each loss term in Eq.4.16 were set to $\lambda_{Adv} = 0.005$ and $\lambda_{TV} = 0.1$.

4.2.3.2 Training Data

Three publicly available datasets were used to train our model, namely ADE20K [80], [81], Places2 [53], and CelebA-HQ [82] datasets. For general image inpainting, we trained our model using the ADE20K and Places2 datasets. For face image inpainting, we trained our model using the CelebA-HQ dataset. We took around 6 days to train either model.

ADE20K dataset. A subset of the ADE20K dataset [80], [81] was used as the training set for the AIM extreme image inpainting challenge [78] of the ECCV 2020. We trained our model on this training set and this dataset was collected for the task of scene parsing and understanding. Therefore, it contains images from different scene categories and is suitable for general image inpainting. There are 10,330 training images in this training set selected by the organizers of [78].

Places2 dataset. This is a commonly used dataset for previous inpainting models as it consists of more than 1.8M training images from 365 scene categories. We randomly selected around half of the validation images, i.e. 17,000 out of 36,500, as our training images. Therefore, together with the ADE20K training set, we had 27,330 training images for our general image inpainting model.

CelebA-HQ dataset. Apart from general image inpainting, we also trained our model for face image inpainting on the CelebA-HQ dataset [82] which contains 30,000 high-quality

face images. Similar to previous methods such as [69], [73], [75] we randomly chose 27,000 images as the training data for our face inpainting model.

4.2.4 Experimental Results

We have thoroughly evaluated our proposed model. We compare our model with state-of-the-art inpainting methods, namely DeepFillv1 [68], GMCNN [69], DeepFillv2 [62], DMFN [73], DeepGIN [75], HiFill [64], and CRFill [76]. Results of the tests show that our model is able to handle images in the wild by testing it on three publicly available datasets: Flickr-Faces-HQ (FFHQ) dataset [91], The Oxford Buildings (Oxford) dataset [92], and Places2 dataset [53].

4.2.4.1 Comparison with Previous Models

We compare our best model against the following state-of-the-art approaches, DeepFillv1, GMCNN, DeepFillv2, DMFN, DeepGIN, HiFill, and CRFill. We directly used their officially provided pre-trained models on CelebA-HQ dataset [82] for face image inpainting on FFHQ, and Places2 training set [53] for general image inpainting on Oxford and Places2 testing sets. Note that authors of DMFN only provided pre-trained model on CelebA-HQ dataset while HiFill and CRFill only trained their models on Places2 dataset.

The FFHQ dataset is similar to the CelebA-HQ dataset and it consists of 70,000 high-quality face images with 1024×1024 resolution. We randomly selected 1,000 images for the testing on this dataset. For the Oxford dataset, it contains 5,062 images of Oxford landmarks with a wide variety of styles. The images include buildings, suburban areas, halls, people, etc. For the Oxford dataset, we randomly chose 523 testing images for comparison. For the Places2 dataset, there are 365 scene categories and 900 testing images per category. Similarly, we also randomly selected 1,000 images from the testing set for comparison.

To test the generalization of different inpainting models, testing images were arbitrarily masked by the three types of masks. All the previous works provided their pre-trained models on 256×256 images. Note also that some models were only trained for the rectangular masks. For fair comparison, we report test results on testing images masked solely by each type of masks and also mixture of the three types of masks. For testing images with size of 256×256 and 512×512, we obtained the inpainting results by a single forward pass. For 1024×1024 images, we divided them into four 512×512 sub-images and obtained the completed sub-images. We then regrouped them to form the completed images with size of 1024×1024.

Quantitative comparisons. Table 4.3 shows the comparisons of the state-of-the-art models and our approach on three datasets with four sets of masked images. The first three sub-tables in Table 4.3 list the test results on FFHQ dataset at three different resolutions, namely 256×256, 512×512 and 1024×1024. We can see that our model is comparable to DeepGIN which focuses on the pixel-wise reconstruction accuracy. Generally, for the evaluation metrics which target at pixel-wise reconstruction accuracy, i.e. PSNR, SSIM [93]

Method	FFHQ(256) Rect. Masks				FFHQ(256) Free-Form Masks				FFHQ(256) Cell-Auto Masks				FFHQ(256) Mixture of 3 Types			
	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS
DeepFillv1	22.70	0.8501	12.80	0.128	20.63	0.7492	10.89	0.273	20.79	0.6406	9.62	0.406	21.34	0.7447	11.92	0.272
GMCNN	24.20	0.8690	10.54	0.105	24.20	0.8400	6.31	0.167	24.88	0.8098	5.29	0.218	22.43	0.8186	10.43	0.189
DeepFillv2	24.30	0.8645	10.47	0.110	24.54	0.8312	6.31	0.182	24.11	0.7690	6.16	0.304	24.32	0.8219	7.83	0.200
DMFN	25.15	0.8783	9.04	0.089	24.99	0.8435	5.89	0.151	20.55	0.6638	9.42	0.358	23.50	0.7937	8.98	0.201
DeepGIN	**25.85**	**0.8896**	**8.37**	**0.086**	**26.48**	**0.8726**	**4.60**	0.114	27.51	0.8621	**3.66**	0.122	**26.57**	**0.8750**	**5.49**	0.107
Ours	25.41	0.8828	8.82	0.088	26.36	0.8714	4.63	**0.109**	**27.53**	**0.8626**	**3.66**	**0.115**	26.40	0.8726	5.58	**0.104**

Method	FFHQ(512) Rect. Masks				FFHQ(512) Free-Form Masks				FFHQ(512) Cell-Auto Masks				FFHQ(512) Mixture of 3 Types			
	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS
DeepFillv1	20.94	0.8523	16.44	0.167	26.99	0.9237	9.18	0.121	21.52	0.6556	8.36	0.447	23.20	0.8116	14.31	0.243
GMCNN	24.09	0.8837	10.50	0.119	33.66	0.9602	3.96	**0.050**	27.64	0.8390	3.57	0.225	24.92	0.8585	18.06	0.162
DeepFillv2	21.84	0.8798	14.31	0.163	32.65	0.9575	4.68	0.074	26.03	0.8132	4.60	0.342	26.91	0.8833	9.70	0.190
DMFN	22.13	0.8743	13.32	0.145	32.87	0.9588	3.89	0.061	20.82	0.6442	8.84	0.434	25.42	0.8310	12.46	0.209
DeepGIN	**25.76**	**0.9044**	**8.44**	**0.116**	34.31	0.9650	3.19	0.058	**28.76**	**0.8711**	**2.97**	0.215	**29.58**	**0.9123**	**6.13**	0.130
Ours	25.33	0.8997	8.92	**0.116**	**34.44**	**0.9656**	**3.14**	0.055	28.75	0.8703	2.99	**0.210**	29.48	0.9105	6.27	**0.127**

Method	FFHQ(1024) Rect. Masks				FFHQ(1024) Free-Form Masks				FFHQ(1024) Cell-Auto Masks				FFHQ(1024) Mixture of 3 Types			
	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS
DeepFillv1	19.92	0.8678	19.23	0.178	33.46	0.9795	10.19	0.045	22.40	0.6744	7.22	0.467	25.29	0.8426	30.87	0.225
GMCNN	20.12	0.8910	19.36	0.160	**41.70**	0.9900	4.65	**0.017**	29.42	0.8625	2.86	**0.250**	27.54	0.8956	47.47	0.161
DeepFillv2	20.70	0.8988	16.96	0.175	40.72	0.9895	5.75	0.026	28.30	0.8497	3.44	0.356	29.85	0.9137	21.10	0.183
DMFN	20.43	0.8916	16.88	0.158	39.80	0.9892	3.59	0.029	21.19	0.6365	8.43	0.488	27.14	0.8424	30.52	0.220
DeepGIN	**25.40**	**0.9145**	**8.87**	0.140	41.34	**0.9907**	**2.74**	0.024	**29.52**	**0.8713**	**2.69**	0.293	**32.15**	**0.9268**	**12.56**	0.149
Ours	24.94	0.9099	9.39	**0.136**	41.34	0.9905	2.75	0.021	29.49	0.8670	2.73	0.296	32.00	0.9238	13.04	**0.148**

Method	Oxford(256) Rect. Masks				Oxford(256) Free-Form Masks				Oxford(256) Cell-Auto Masks				Oxford(256) Mixture of 3 Types			
	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS
DeepFillv1	20.70	0.8111	15.92	0.155	17.57	0.6983	15.90	0.292	18.78	0.5995	12.11	0.393	18.85	0.7085	16.14	0.273
GMCNN	18.97	0.7905	19.51	0.166	22.31	0.7968	8.04	0.187	22.66	0.7604	6.82	0.242	21.12	0.7815	11.07	0.196
DeepFillv2	20.37	0.8120	16.60	0.153	21.79	0.7887	8.61	0.197	22.38	0.7553	7.08	0.252	21.26	0.7843	10.78	0.199
DeepGIN	**22.43**	**0.8320**	**12.98**	0.148	23.43	0.8123	6.92	0.175	23.81	0.7891	5.73	0.197	**22.94**	0.8092	8.66	0.175
HiFill	19.90	0.7809	18.11	0.178	20.03	0.7084	12.30	0.294	17.29	0.4830	17.16	0.495	19.16	0.6693	17.36	0.311
CRFill	20.99	0.8167	14.85	0.148	22.32	0.8034	7.73	0.187	22.36	0.7672	6.67	0.230	21.69	0.7951	9.84	0.188
Ours	22.25	0.8286	13.20	**0.143**	**23.46**	**0.8167**	**6.81**	**0.163**	**23.97**	**0.7978**	**5.58**	**0.179**	**22.94**	**0.8123**	**8.59**	**0.163**

Method	Oxford(512) Rect. Masks				Oxford(512) Free-Form Masks				Oxford(512) Cell-Auto Masks				Oxford(512) Mixture of 3 Types			
	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS
DeepFillv1	20.32	0.8226	16.80	0.157	22.73	0.9058	15.20	0.117	19.06	0.6108	11.17	0.414	20.62	0.7666	17.41	0.241
GMCNN	16.20	0.7853	27.78	0.180	29.73	0.9435	6.32	0.062	23.80	0.7751	5.71	0.255	23.08	0.8274	14.46	0.174
DeepFillv2	19.68	0.8230	17.89	0.159	29.07	0.9407	6.93	0.066	23.58	0.7744	5.89	0.270	23.87	0.8398	11.77	0.173
DeepGIN	**22.26**	**0.8518**	**13.21**	0.161	30.27	0.9455	5.41	0.078	24.52	0.7928	5.11	0.288	25.40	0.8577	9.41	0.184
HiFill	19.53	0.7980	18.79	0.174	25.54	0.8939	12.20	0.130	14.72	0.3784	26.95	0.612	19.38	0.6658	33.15	0.325
CRFill	20.48	0.8273	15.75	**0.151**	29.83	0.9469	5.35	0.062	23.75	0.7879	5.43	0.241	24.41	0.8477	10.43	0.159
Ours	22.16	0.8499	13.36	0.155	**30.91**	**0.9512**	**4.82**	**0.050**	**25.30**	**0.8162**	**4.56**	**0.186**	**25.70**	**0.8643**	**9.22**	**0.136**

Method	Places2(256) Rect. Masks				Places2(256) Free-Form Masks				Places2(256) Cell-Auto Masks				Places2(256) Mixture of 3 Types			
	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS
DeepFillv1	20.94	0.8154	15.77	0.150	18.06	0.7050	15.06	0.288	19.09	0.6093	11.74	0.386	19.36	0.7132	15.65	0.271
GMCNN	19.28	0.7954	19.12	0.159	22.31	0.7944	8.18	0.179	22.80	0.7590	6.80	0.229	21.51	0.7846	11.10	0.187
DeepFillv2	20.60	0.8176	16.39	0.145	21.92	0.7908	8.61	0.187	22.64	0.7594	6.97	0.239	21.74	0.7906	10.76	0.188
DeepGIN	**22.73**	**0.8350**	**12.79**	0.141	23.46	0.8089	7.07	0.172	24.01	0.7871	5.75	0.190	23.45	0.8112	8.66	0.166
HiFill	20.12	0.7829	18.07	0.176	20.06	0.7030	12.46	0.295	17.27	0.4720	17.49	0.498	19.31	0.6611	17.84	0.316
CRFill	21.21	0.8198	14.98	0.141	22.37	0.8002	7.87	0.180	22.52	0.7671	6.70	0.219	22.09	0.7970	9.98	0.178
Ours	22.53	0.8320	13.03	**0.136**	**23.55**	**0.8138**	**6.92**	**0.159**	**24.18**	**0.7964**	**5.59**	**0.173**	**23.46**	**0.8148**	**8.59**	**0.155**

Table 4. 3 Quantitative comparisons on FFHQ, Oxford and Places2 datasets. The best and second-best results are in **bold** and blue respectively

and mean L1 error, we are comparable to DeepGIN. Note that higher PSNR, SSIM and smaller

L1 err. (%) mean better pixel-wise reconstruction accuracy. For the metric to estimate the visual quality of the output, i.e. Learned Perceptual Image Patch Similarity (LPIPS) [95], the smaller the better. Our model performs slightly better than DeepGIN on average.

To ensure the usefulness of the completed faces in other high-level tasks such as face verification and recognition, we used deepface [96] to obtain the completed face verification rates of different methods. Figure 4.10 shows the verification rates using different threshold settings. We can see that both DeepGIN and ours can offer better verification rates of the completed faces as compared with other existing methods. This is an interesting way to evaluate the completed faces as the completed faces could be used for other high-level computer vision tasks but not just for viewing.

The bottom three sub-tables in Table 4.3 show the test results on Oxford and Places2 datasets. Note that general image inpainting is much challenging compared with face image inpainting. For Oxford dataset, we report the test results for two resolutions, 256 and 512 while we report the test results on Places2 dataset for 256×256 resolution. It is obvious that our model outperforms other inpainting methods in most cases.

Qualitative comparisons. Figure 4.9 displays the test results on FFHQ dataset. Each row shows an example of the test results. From left to right: the first column shows different masked images. The second to the seventh columns display the completed faces by different methods, and the last column shows the reference ground truth. It can be seen that only DeepGIN and ours can achieve satisfactory visual quality without obvious visual artifacts on large masked regions in general. Compared to DeepGIN, our model offers more visual details such as the textures of the hair and the continuity of the glasses in the first and second examples respectively. For other examples, our model also provides the inpainting results with better global semantics and local textures. Note that DeepGIN is L1-oriented, hence blurry inpainting

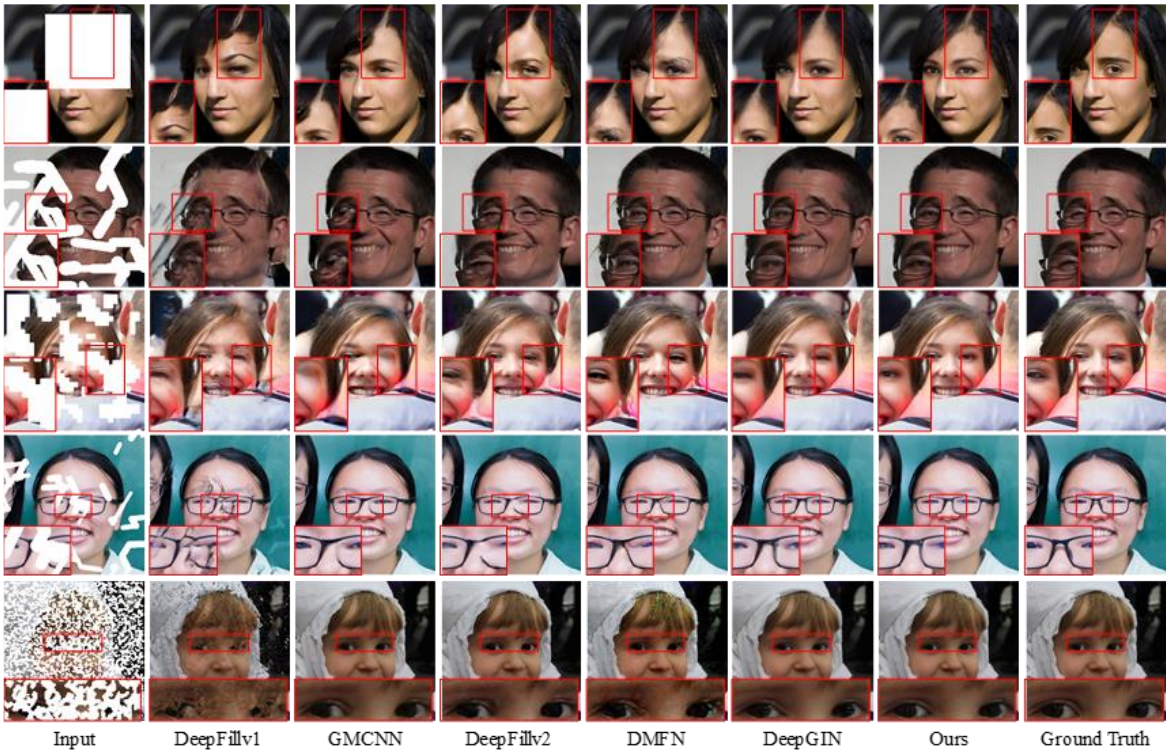


Figure 4. 9 Comparisons of test results on FFHQ dataset. Each row shows an example by using various methods. Zoom in for a better view.

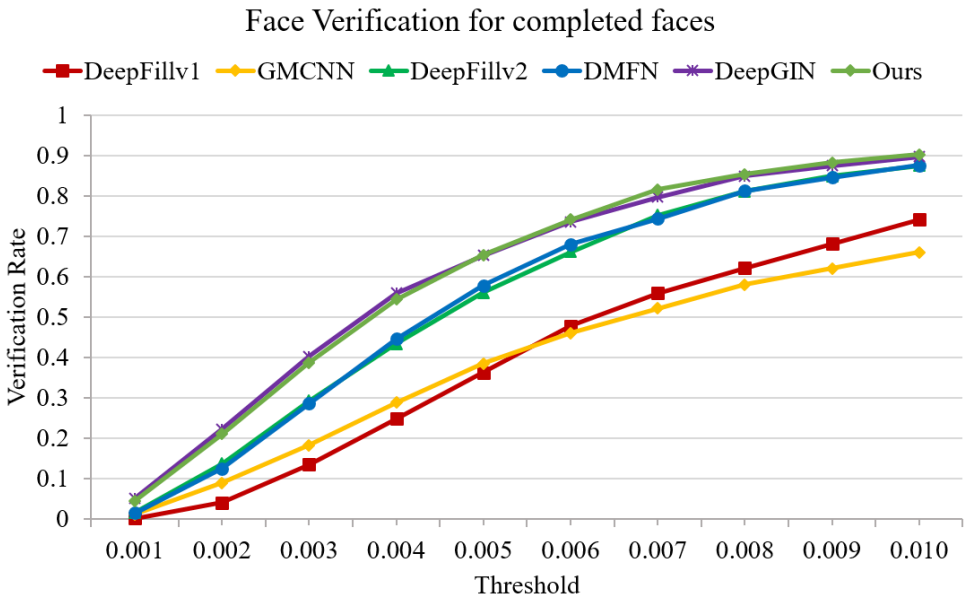


Figure 4. 10 Face verification rates for completed faces by various approaches under different threshold settings. Vertical axis shows the verification rate which ranges from 0 to 1; horizontal axis shows different threshold values for verification, from 0.001 to 0.01.

results are observed. On the other hand, we achieve a better balance between pixel-wise accuracy and visual quality compared to the existing methods.

Figure 4.11 shows the test results on Oxford and Places2 datasets. To show the importance of balancing pixel-wise accuracy and visual quality such that the inpainting results can be used for other high-level computer vision tasks, we also offer the predicted semantic segmentation test results as computed by a well pre-trained segmentation network [97]. Each pair of two rows shows an example of the inpainting results by various methods with the corresponding predicted segmentation results. From left to right: the first column displays different masked images and the corresponding mask images. The second to the eighth columns show the inpainting results and the corresponding segmentation results, and the last column displays the reference ground truth images and the predicted segmentation results. Generally, only DeepGIN and ours can provide plausible global semantics with no severe visual artifacts for the cases of rectangular masks, see for example, the skeleton of the building in the first example. For the other two types of masks: DeepFillv2, DeepGIN, CRFill and ours can give satisfactory visual quality. For example, we can see the bicycle wheel and the window grille in the fifth (the 9^{th} and 10^{th} rows) and sixth (the last two rows) examples respectively in Figure 4.11. For the predicted segmentation results, it is clear that the test results as computed by DeepGIN and ours are semantically closer to the predicted segmentation results using the reference ground truth images compared to that of the other existing methods.

We also compare our model with other inpainting models on real-world object removal examples and the results are shown in Figure 4.12. We can see that CRFill and ours can offer more satisfactory inpainting results compared with other methods in general. When we zoom in the inpainting results to see the textures of the grass field (the last row), we can see that ours is the most consistent with the textures of the grass field from the known areas. By performing object removal for the input images, we can have cleaner images as input to perform other computer vision tasks.

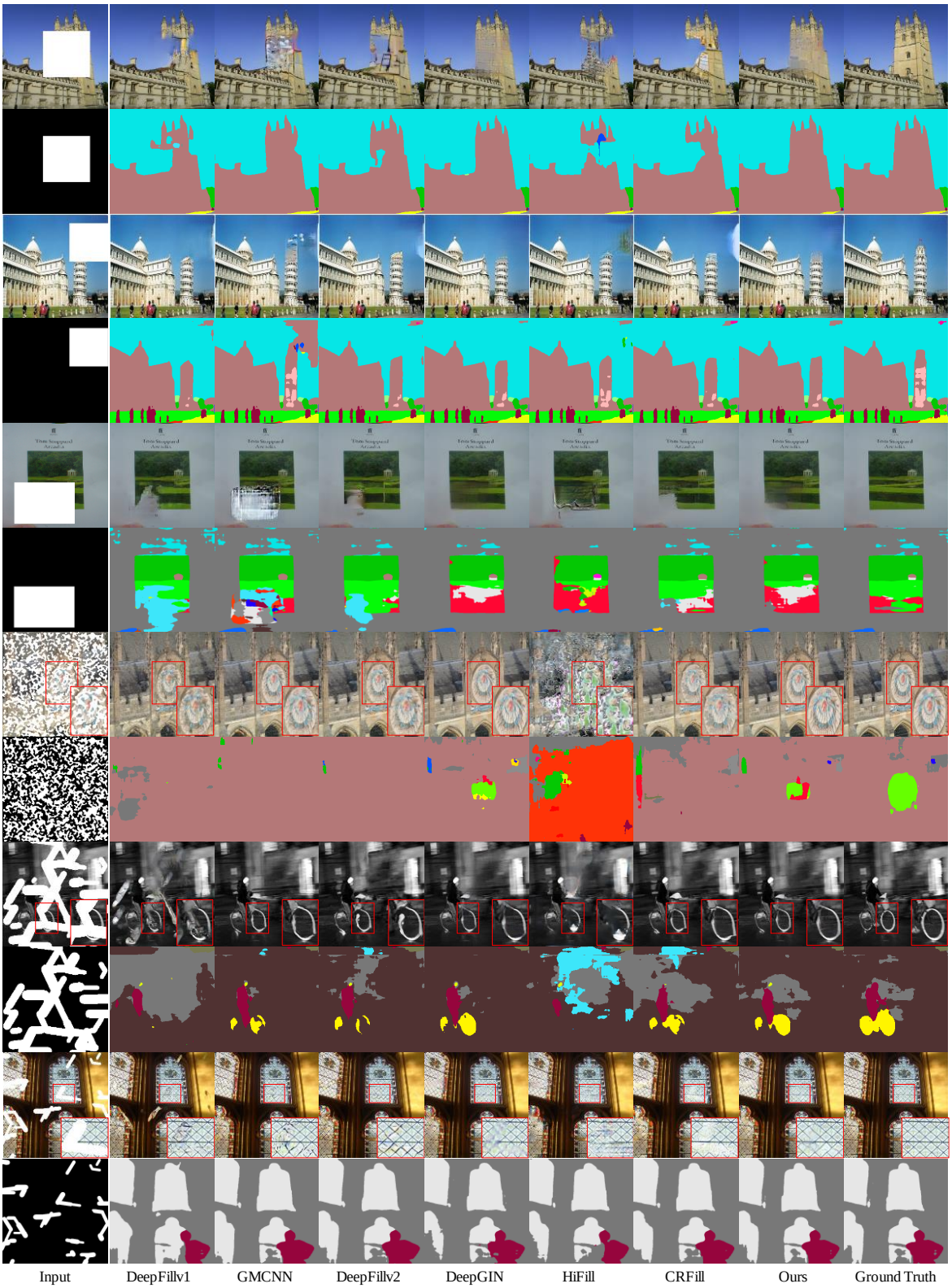


Figure 4. 11 Comparisons of test results on Oxford and Places2 datasets. Every two rows show an example with the corresponding segmentation results. Various types of masks are considered, and the mask images are shown at the first column of every two rows. Please zoom in for a better view of the inpainting results and the corresponding segmentation results

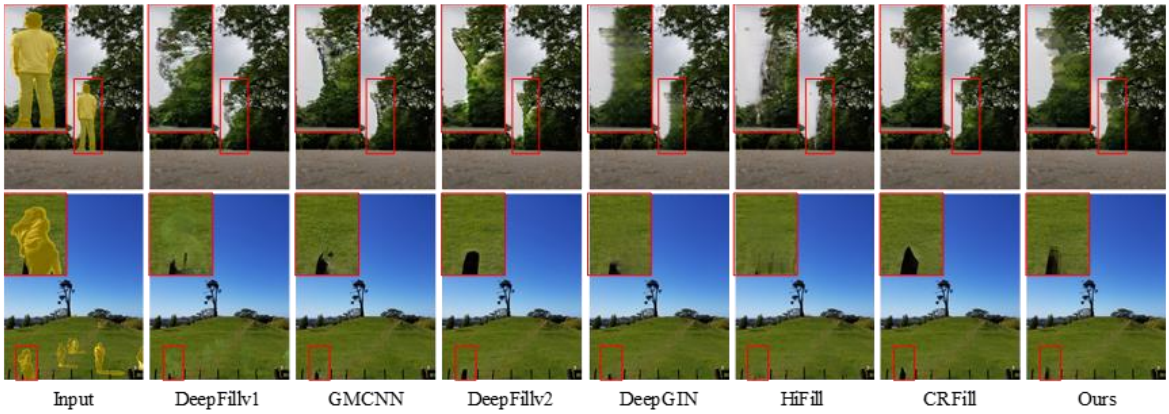


Figure 4. 12 Comparisons of inpainting results on real-world object removal examples. Photos were captured by the author of this thesis. Please zoom in for a better view.

4.2.4.2 Model Analysis

In this sub-section, we evaluate the effectiveness of our auxiliary contextual attention learning branch (AT) and multi-dilation fusion block (MDFB). We also study the effects of using the middle layer and higher semantic layer for computing the perceptual loss. Our baseline is denoted as SD-Relu3 which replaces the 4 MDFBs of our generator (i.e. the 4 orange blocks located at the middle of *G* (light blue shaded area) as shown in Figure 4.6) as 4 standard dilated convolutions with dilation rates = 2, 4, 8, and 16 respectively. We used VGG16 relu3_2 middle layer for computing the perceptual loss. This baseline was trained without using AT and this network architecture is the same as the coarse generator used in [62], [68], [76]. AT-SD-Relu3 represents the baseline was trained with the proposed AT. AT-SDFB-Relu3 means that the 4 standard dilated convolutions with dilation rates = 2, 4, 8, and 16 were replaced by 4 MDFBs which used 4 different dilation rates, namely 2, 4, 8, 16. We named this setting as single-dilation fusion block (SDFB). This means that the six parallel dilated convolutions with six different dilation rates of the first SDFB were replaced by six identical dilation rate = 2. The second, third and fourth SDFB used identical dilation rates = 4, 8, and 16 respectively. AT-MDFB-Relu3 and -Relu4 represent our final model and the only difference is that one used VGG16 relu3_2 middle layer for computing the perceptual loss and another one used VGG16

relu4_2 high-level semantic layer. We conducted our model analysis on general image inpainting, i.e. Oxford and Places2 datasets, using the same set of testing images as mentioned in the previous sub-section.

Method	Oxford(256) Rect. Masks				Oxford(256) Free-Form Masks				Oxford(256) Cell-Auto Masks				Oxford(256) Mixture of 3 Types			
	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS
SD-Relu3	21.15	0.8077	15.22	0.161	22.39	0.7845	8.03	0.211	22.84	0.7542	6.63	0.250	21.87	0.7801	10.07	0.207
AT-SD-Relu3	21.29	0.8107	14.87	0.156	22.60	0.7894	7.80	0.198	23.01	0.7616	6.45	0.230	22.02	0.7853	9.82	0.195
AT-SDFB-Relu3	21.81	0.8201	13.86	0.145	23.12	0.8076	7.15	0.168	23.69	0.7887	5.81	0.185	22.58	0.8031	8.99	0.167
AT-MDFB-Relu3	21.99	0.8217	13.59	**0.143**	23.27	0.8117	6.97	**0.162**	23.86	0.7934	5.67	**0.177**	22.74	0.8067	8.78	**0.162**
AT-MDFB-Relu4	**22.25**	**0.8286**	**13.20**	**0.143**	**23.46**	**0.8167**	**6.81**	0.163	**23.97**	**0.7978**	**5.58**	0.179	**22.94**	**0.8123**	**8.59**	0.163

Method	Oxford(512) Rect. Masks				Oxford(512) Free-Form Masks				Oxford(512) Cell-Auto Masks				Oxford(512) Mixture of 3 Types			
	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS
SD-Relu3	21.09	0.8344	15.23	0.167	29.37	0.9387	6.14	0.095	23.53	0.7610	5.88	0.346	24.43	0.8379	10.96	0.212
AT-SD-Relu3	21.25	0.8367	14.88	0.163	29.54	0.9400	5.98	0.090	23.71	0.7664	5.73	0.331	24.60	0.8411	10.64	0.204
AT-SDFB-Relu3	21.83	0.8443	13.87	0.155	30.27	0.9460	5.40	0.075	24.45	0.7922	5.16	0.280	25.24	0.8549	9.67	0.178
AT-MDFB-Relu3	21.96	0.8451	13.65	**0.154**	30.79	0.9502	4.91	**0.050**	25.19	0.8118	4.65	**0.185**	25.55	0.8597	9.38	**0.134**
AT-MDFB-Relu4	**22.16**	**0.8499**	**13.36**	0.155	**30.91**	**0.9512**	**4.82**	**0.050**	**25.30**	**0.8162**	**4.56**	0.186	**25.70**	**0.8643**	**9.22**	0.136

Method	Places2(256) Rect. Masks				Places2(256) Free-Form Masks				Places2(256) Cell-Auto Masks				Places2(256) Mixture of 3 Types			
	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS	PSNR	SSIM	L1 err. (%)	LPIPS
SD-Relu3	21.40	0.8105	15.13	0.157	22.37	0.7800	8.20	0.211	22.85	0.7479	6.74	0.247	22.23	0.7808	10.18	0.203
AT-SD-Relu3	21.54	0.8142	14.78	0.150	22.56	0.7848	7.99	0.197	23.05	0.7560	6.53	0.227	22.42	0.7862	9.92	0.190
AT-SDFB-Relu3	22.05	0.8235	13.77	0.138	23.14	0.8039	7.30	0.165	23.84	0.7866	5.86	0.179	23.05	0.8052	9.05	0.160
AT-MDFB-Relu3	22.30	0.8258	13.36	**0.136**	23.43	0.8098	7.03	**0.155**	24.13	0.7941	5.64	**0.166**	23.32	0.8106	8.74	**0.152**
AT-MDFB-Relu4	**22.53**	**0.8320**	**13.03**	**0.136**	**23.55**	**0.8138**	**6.92**	0.159	**24.18**	**0.7964**	**5.59**	0.173	**23.46**	**0.8148**	**8.59**	0.155

Table 4. 4 Model analysis of our model on Oxford and Places2 datasets. The best and second-best results are in **bold** and blue respectively

Quantitative comparisons. The quantitative results of our model analysis are listed in Table 4.4. Similarly, higher PSNR, SSIM and smaller L1 err. (%) imply better pixel-wise reconstruction accuracy while smaller LPIPS means better estimated visual quality of the completed images. It is clear that AT-SD-Relu3 offers better results as compared with SD-Relu3 in terms of both pixel-wise accuracy and visual quality in all cases. This means that the auxiliary contextual attention learning branch is useful to boost the inpainting performance. We can also observe that SDFB offers better results as compared with SD while MDFB further improves the inpainting performance compared to SDFB. This shows the improvement on using the ideas of feature fusion and multi-dilation. For the choice of feature layer for computing the perceptual loss, we can see that using the middle layer (Relu3) leads to better

estimated visual quality, i.e. smaller LPIPS, while using the higher semantic layer (Relu4) results in better pixel-wise reconstruction accuracy, i.e. higher PSNR, SSIM, and smaller L1 err. in most cases. As the improvement in the estimated visual quality is relatively slighter than that of the pixel-wise reconstruction accuracy, we chose AT-MDFB-Relu4 as our final model and compared it with the state-of-the-art models as mentioned in the previous sub-section.

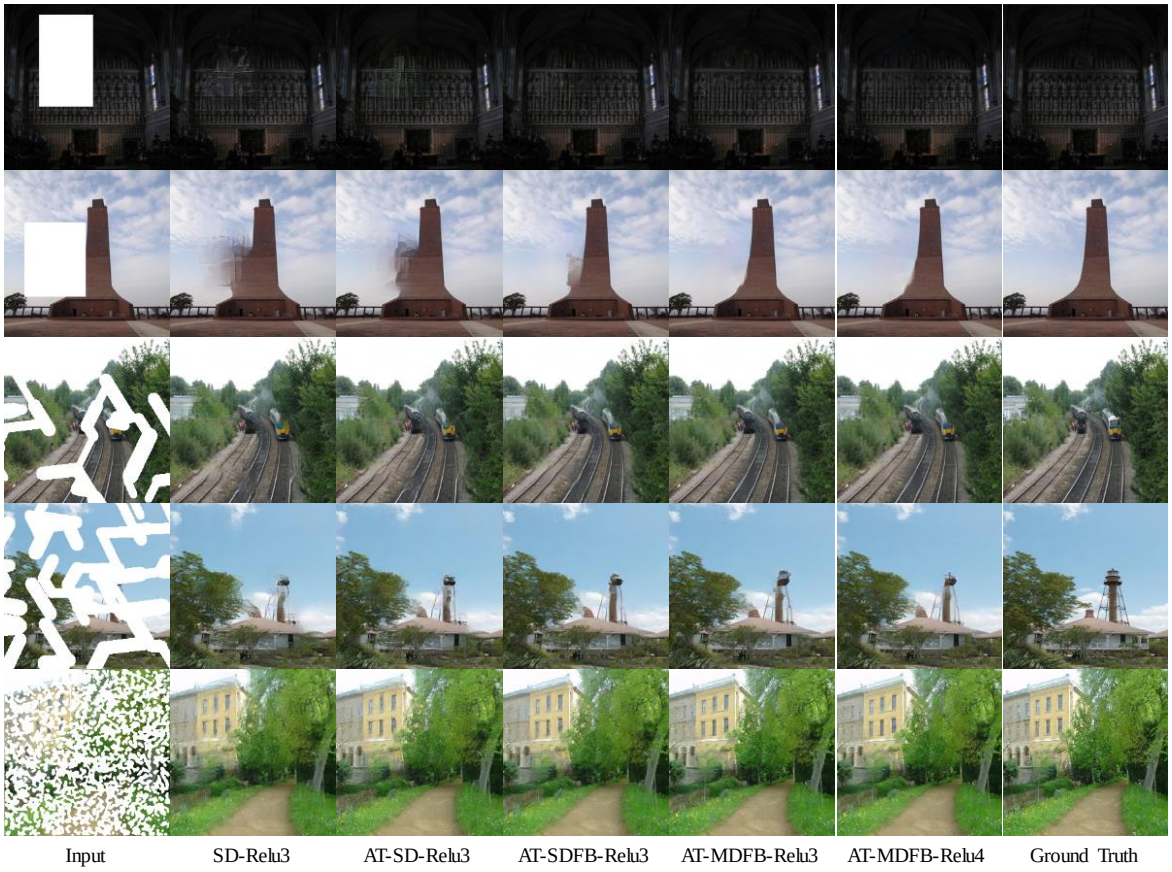


Figure 4. 13 Comparisons of test results of the variants of our proposed model on Oxford and Places2 datasets. Each row shows an example. Zoom in for a better view.

Qualitative comparisons. Figure 4.13 shows the comparisons of five variants of our proposed model on Oxford and Places2 datasets. Obviously, the visual quality of the completed images without using both the AT and MDFB is unsatisfactory. For AT-MDFB-Relu3 and AT-MDFB-Relu4, these two variants offer similar visual quality. For example, please see the carved wall and the train track in the first and the third rows respectively. The two variants can

accurately connect the train track with good continuity while the other three variants cannot do so.

4.2.5 Conclusion on Inpainting with Global Semantics

In Section 4.2, we have proposed a general image inpainting model which achieves a better balance of pixel-wise reconstruction accuracy and visual quality of the inpainting results, as compared with the existing inpainting models. We design a multi-dilation fusion block and an auxiliary attention learning branch for enhancing the global semantics and local textures of the inpainting results. Our model analysis has demonstrated the effectiveness of our proposed auxiliary attention learning branch and multi-dilation fusion block. We have achieved inpainting performance better than the state-of-the-art models in term of both visual quality and semantic correctness. We have also demonstrated the possible use of the inpainting results for other high-level computer vision tasks, such as face verification and semantic segmentation. For future development, we can try reference-based image inpainting to further boost the local textures of the generated content. We hope that finer details can be generated by attaching an extra common reference image with full information.

4.3 Chapter Summary

In this chapter, we discussed our proposed inpainting models which target at a better balance between pixel-wise reconstruction accuracy and visual quality. We emphasize the use of the inpainting results to other high-level computer vision tasks like recognition, verification, and segmentation. Our experimental results demonstrate the feasibility in using our inpainting results for other high-level computer vision tasks. Future work on applying synthetic results to other tasks will be done in order to improve the generalization of our algorithms to real-world situations.

In our proposed inpainting models, we focus on two techniques for improving the inpainting performance. The first technique is on dilated convolution which offers larger receptive fields for a model to see more distant spatial locations without an obvious increase in the number of parameters. The second technique is on self-attention which allows our models to borrow realistic information from the valid pixels to reconstruct and refine the missing and generated pixels respectively. Our experimental results show that we can achieve the state-of-the-art inpainting performance in terms of both the visual quality and high-level semantic accuracy.

We finally also provide some examples of quality scene reconstruction by using inpainting models to perform object removal. The experiment results demonstrate that better scenes could be obtained by removing those unwanted objects. In the future, the proposed inpainting models would be used as a pre-processing module in real-life recognition system to boost the discriminative power of the input images.

Chapter 5 Conclusion

In this thesis, we have first proposed a fast monocular visual place recognition for situations with varying speed situations and dynamic lighting environments using conventional machine learning techniques. We define key frames in a video sequence as the places where are obviously different from the other frames in the same sequence, hence they can be easily recognized. We identify and analyze key frames using HOG features and low-resolution whole frame normalized descriptors. We successfully represent a key frame by only few but effective feature patches with variable patch sizes. To achieve efficient recognition, we report the recognition results through comparing low-resolution whole frame descriptors with our idea of tube of frames. Note that the estimation of the next best match based on previous results helps to define the search range for the current feature matching. The final recognition result is derived from both the estimation and the current calculation. For the idea of tube of frames, it emphasizes the temporal logic constraints on the movement of a vehicle in the form of the previous recognition results. We always travel along a path gradually without sudden jump. We also propose a two-stage key frame recognition to resolve the possible accumulated deviation from the ground truth without an obvious increase in the computational cost of our proposed method. Detailed patch-based matching will only be activated if there is a need. When a key frame is recognized, the corresponding location of the key frame will be used to reset the location of the tube such that the accumulated deviation can be cleared. Our proposed method achieves a high $F1$ score and is less sensitive to the length of sequence. The method is also a good choice when there is a need to balance the recognition performance and the computational cost, especially for resource-limited and/or battery-powered devices.

We also further develop our concept of tube of frames together with the recent deep features. To allow the use of CNNs in visual place recognition, we study a lightweight CNN model and enhance an automatic training data generation module. A lightweight CNN model

can more efficiently extract deep features and the data generation module can generate a wide range of labelled data, for which variations in appearance, seasons, lighting environments, and viewpoints are all included. We first tackle the problem of unknown starting location of a vehicle with the deep features and our dynamic tubing strategy. Our proposed method gives the initialization result by adaptively considering a number of consecutive incoming frames based on the amount of new information from the incoming frames and the confidence of the frame matching. We attain a high confident initialization with 80% accuracy compared to 57.1% without the proposed tubing strategy, 47.1% and 32.9% of two AlexNet conv3 feature-based approaches, pre-trained on place-centric dataset and object-centric dataset respectively.

We also extend our initialization module by integrating it into an efficient tubing strategy-based place recognition method. After the initialization, similar idea of the tubing strategy is applied to the recognition module. We propose the use of weighted sums of the similarity scores based on the comparison of the consecutive query frames to obtain the final match pairs. Similarly, the search spaces of the current incoming query frame are defined by the previous match pairs, hence the searching complexity can be reduced. Our experimental results show that the proposed efficient recognition method offers a satisfactory $F1$ score of 0.724 compared to the second-best method which achieves 0.658 and is a conventional sequence-based method. Our tubing strategy is also faster than the standard linear full search strategy by a factor of 2.15, i.e., 12.4 milliseconds compared to 26.7 milliseconds using a standard commodity CPU device.

To further improve the discriminative power of the deep features for places under different conditions, we also study the topic of quality scene reconstruction. We would like to remove dynamic and unwanted objects like pedestrians and vehicles in every incoming frame such that we can extract features from clean images. Therefore, we present a deep generative inpainting network named DeepGIN. We enlarge the receptive fields of our model by

proposing a Spatial Pyramid Dilation (SPD) residual block to cover more distant spatial locations. The ability of a model to observe distant spatial locations through dilation convolutions is important to inpainting especially when the missing regions are large. We also suggest to use Multi-Scale Self-Attention (MSSA) blocks to enhance the importance of self-similarity such that the generated pixels can be further improved by using the valid pixels. Furthermore, Back Projection (BP) strategy is applied to improve the alignment of the generated and valid pixels. Our model attains a better balance of pixel-wise reconstruction accuracy and visual quality compared to the state-of-the-art methods. We extend our DeepGIN by designing a Multi-Dilation Fusion Block (MDFB) and an auxiliary attention learning branch for improving the global semantics and local textures of the inpainting results. The experimental results have demonstrated the effectiveness of the proposed MDFB and the auxiliary attention learning branch. Our model achieves the state-of-the-art performance in terms of both visual quality and semantic accuracy. We have also shown the feasibility of using our inpainting results for other high-level computer vision tasks like semantic segmentation and face verification.

For future development, it is a promising research direction to build a comprehensive localization system. Our proposed inpainting models can be used as a pre-processing module to remove unwanted objects and/or occlusions for getting clean input images. Then, we can extract more robust deep features from these clean input images for feature matching and recognition. Finally, our proposed tubing strategy can also be used as a post-processing step to further enhance the recognition performance. Integrating the tubing strategy into CNNs would also be a good direction to study. An online learning strategy to continuously update the place database would be important to develop a life-long localization and recognition system.

Chapter 6 References

[1] R. Arandjelović and A. Zisserman, "All about VLAD," *in Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Portland, OR, USA, Jun. 2013, pp. 1-8.

[2] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," *in Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.* (*CVPR*), Las Vegas, NV, USA, Jun. 2016, pp. 1-17.

[3] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1437-1451, Jun. 2018.

[4] Guillaume Bresson, Zayed Alsayed, Li Yu, and Sébastien Glaser, "Simultaneous Localization And Mapping: A Survey of Current Trends in Autonomous Driving," *IEEE Trans. on Intelligent Vehicles*, vol.2, no.3, pp.194-220, Sept. 2017.

[5] 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: Towards the Robust-Perception Age," *IEEE Trans. on Robotics*, vol.32, no.6, pp.1309-1332, Dec. 2016.

[6] Yongliang Qiao, Cindy Cappelle, and Yassine Ruichek, "Visual Localization across Seasons using Sequence Matching based on Multi-Feature combination," *Journal of Sensors*, vol. 17, no. 11, pp. 2442-2463, Oct. 2017.

[7] Michael J. Milford and Gordon F. Wyeth, "SeqSLAM: Visual Route-Based Navigation for Sunny Summer Days and Stormy Winter Nights," *Proceedings, IEEE International Conference on Robotics and Automation (ICRA)*, Saint Paul, MN, USA, pp. 1643-1649, May 2012.

[8] Michael J. Milford, Felix Schill, Peter Corke, Robert Mahony, and Gordon Wyeth, "Aerial SLAM with a Single Camera Using Visual Expectation," *Proceedings, IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, pp. 2506-2512, May 2011.

[9] Niko Sünderhauf and Peter Protzel, "BRIEF-Gist - Closing the Loop by Simple Means," Proceedings, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA, USA, pp. 1234-1241, Sep. 2011.

[10] Mark Cummins and Paul Newman, "Highly Scalable Appearance-Only SLAM - FAB-MAP 2.0," *Proceedings, Conference on Robotics: Science and Systems (RSS)*, Seattle, Washington, United States of America, pp. 39-46, 28 Jun. to 1 Jul. 2009.

[11] Michael J. Milford, Walter Scheirer, Eleonora Vig, Arren Glover, Oliver Baumann, Jason Mattingley, and David Cox, "Condition-Invariant, Top-Down Visual Place Recognition," Proceedings, *IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, pp. 5571-5577, May 2014.

[12] Meng Yao, Wan-Chi Siu and Ke-Bin Jia, "Learning-based Scene Recognition with Monocular Camera for Light-Rail System," *Proceedings, IEEE International*

Conference on Industrial Electronics for Sustainable Energy Systems (IESES), Hamilton, New Zealand, pp.230-236, 30 Jan. to 2 Feb. 2018.

[13] Niko Sünderhauf, Ssareh Shirzai, Feras Dayoub, Ben Upcroft, and Michael Milford, "On the Performance of ConvNet Features for Place Recognition," *Proceedings, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, pp. 4297-4304, Sept. 2015.

[14] Tayyab Naseer, Wolfram Burgard and Cyrill Stachniss, "Robust Visual Localization Across Seasons," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 289-302, Apr. 2018.

[15] Fei Han, Xue Yang, Yiming Deng, Mark Rentschler, Dejun Yang, and Hao Zhang, "SRAL: Shared Representative Appearance Learning for Long-Term Visual Place Recognition," *IEEE Robotics and Automation Letters (RAL)*, vol. 2, no. 2, pp. 1172-1179, Apr. 2017.

[16] Chup-Chung Wong, Wan-Chi Siu, Paul Jennings, Stuart Barnes, and Bernard Fong, "A Smart Moving Vehicle Detection System Using Motion Vectors and Generic Line Features," *IEEE Transactions on Consumer Electronics*, vol. 61, no. 3, pp. 384-392, Aug. 2015.

[17] Xue-Fei Yang and Wan-Chi Siu, "Vehicle Detection under Tough Conditions using Prioritized Feature Extraction with Shadow Recognition," *Proceedings, IEEE 22nd International Conference on Digital Signal Processing (DSP)*, London, UK, pp. 1-5, Aug. 2017.

[18] Hoi-Kok Cheung, Wan-Chi Siu, Steven Lee, Lawrence Poon, and Chiu-Shing Ng, "Accurate Distance Estimation Using Camera Orientation Compensation Technique for Vehicle Driver Assistance System," *Proceedings, IEEE International Conference on Consumer Electronics (ICCE)*, Las Vegas, NV, USA, pp. 231-232, Jan. 2012.

[19] D. Nistér, O. Naroditsky, and J. Bergen, "Visual Odometry," *Proceedings, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Washington, DC, USA, pp. 652-659, 27 Jun. to 2 Jul. 2004.

[20] Chuanqi Cheng, Xiangyang Hao, Zhenjie Zhang, and Mandan Zhao, "Monocular Visual Odometry Based on Optical Flow and Feature Matching," *Proceedings, IEEE the 29th Conference on Chinese Control And Decision (CCDC)*, Chongqing, China, pp. 4554-4558, May 2017.

[21] Haifeng Li, Hongpeng Wang, and Jingtai Liu, "Monocular Visual Odometry Using Vertical Lines in Urban Area," *Proceedings, IEEE the 32nd Conference on Chinese Control (CCC)*, Xi'an, China, pp. 5676-5681, Jul. 2013.

[22] Zhenwei Shi, Zhengxia Zou, and Changshui Zhang, "Real-Time Traffic Light Detection With Adaptive Background Suppression Filter," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 3, pp. 690-700, Mar. 2016.

[23] Sanjay Saini, S. Nikhil, Krishna Reddy Konda, Harish S Bharadwaj, and N. Ganeshan, "An Efficient Vision-Based Traffic Light Detection and State Recognition for

Autonomous Vehicles," *IEEE Intelligent Vehicles Symposium (IV)*, Los Angeles, CA, USA, pp. 606-611, Jun. 2017.

[24] Saleh Aly, Daisuku Deguchi, and Hiroshi Murase, "Blur-invariant Traffic Sign Recognition Using Compact Local Phase Quantization," *Proceedings, IEEE International Conference on Intelligent Transportation Systems (ITSC)*, The Hague, Netherlands, pp. 821-827, Oct. 2013.

[25] Vidyagouri B. Hemadri and Umakant P. Kulkarni, "Road Sign Detection and Recognition in Adverse Case using Pattern Matching," *Proceedings, International Conference & Workshop on Advanced Computing (ICWAC)*, Mumbai, India, pp. 49-53, Feb. 2013.

[26] Hao Wu and Wan-Chi Siu, "Real Time Railway Extraction By Angle Alignment Measure," *Proceedings, IEEE International Conference on Image Processing (ICIP)*, Quebec City, QC, Canada, pp. 4560-4564, Sept. 2015.

[27] Hunjae Yoo, Ukil Yang, and Kwanghoon Sohn, "Gradient-Enhancing Conversion for Illumination-Robust Lane Detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1083-1094, Sept. 2013.

[28] Mark Cummins and Paul Newman, "FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647-665, Jun. 2008.

[29] Jianliang Zhu, Yunfeng Ai, Bin Tian, Dongpu Cao, and Sebastian Scherer, "Visual Place Recognition in Long-term and Large-scale Environment based on CNN Feature," *IEEE Intelligent Vehicles Symposium (IV)*, Changshu, Suzhou, China, pp. 1679-1685, Jun. 2018.

[30] Edward Pepperell, Peter I. Corke, and Michael J. Milford, "All-Environment Visual Place Recognition with SMART," *Proceedings, IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, pp. 1612-1618, May 2014.

[31] Roberto Arroyo, Pablo F. Alcantarilla, Luis M. Bergasa, and Eduardo Romera, "Towards Life-Long Visual Localization using an Efficient Matching of Binary Sequences from Images," *Proceedings, IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, Washington, pp. 6328-6335, May 2015.

[32] Roberto Arroyo, Pablo F. Alcantarilla, Luis M. Bergasa, and Eduardo Romera, "OpenABLE: An Open-source Toolbox for Application in Life-Long Visual Localization of Autonomous Vehicles," *Proceedings, IEEE International Conference on Intelligent Transportation Systems (ITSC)*, Rio de Janeiro, Brazil, pp. 965-970, Nov. 2016.

[33] Sayem Mohammad Siam, and Hong Zhang, "Fast-SeqSLAM: A Fast Appearance Based Place Recognition Algorithm," *Proceedings, IEEE International Conference on Robotics and Automation (ICRA)*, Singapore, pp. 5702-5708, May 2017.

[34] Chu-Tak Li, and Wan-Chi Siu, "Fast Monocular Vision-based Railway Localization for Situations with Varying Speeds," *Proceedings, APSIPA Annual Summit and Conference 2018 (APSIPA-ASC 2018)*, Hawaii, USA, pp. 2006-2013, Nov. 2018.

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Proceedings, the 25th International Conference on Neural Information Processing Systems (NIPS)*, Lake Tahoe, USA, pp. 1097-1105, Dec. 2012.

[36] Nate Merrill and Guoquan Huang, “Lightweight Unsupervised Deep Loop Closure,” *Proceedings, Conference on Robotics: Science and Systems (RSS)*, Pittsburgh, Pennsylvania, USA, pp. 1-9, Jun. 2018.

[37] Roberto Arroyo, Pablo F. Alcantarilla, Luis M. Bergasa, and Eduardo Romera, “Fusion and Binarization of CNN Features for Robust Topological Localization across Seasons,” *Proceedings, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, South Korea, pp. 4656-4663, Oct. 2016.

[38] Tayyab Naseer, Wolfram Burgard, and Cyrill Stachniss, “Robust Visual Localization Across Seasons,” *IEEE Trans. on Robotics*, vol.34, no.2, pp. 289-302, Apr. 2018.

[39] Muneeb Shahid, Tayyab Naseer, and Wolfram Burgard, “DTLC: Deeply Trained Loop Closure Detections for Lifelong Visual SLAM,” *Proceedings, Workshop on Visual Place Recognition, Conference on Robotics: Science and Systems (RSS)*, Ann Arbor, MI, USA, pp. 1-8, Jun. 2016.

[40] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol, “Extracting and Composing Robust Features with Denoising Autoencoders,” *Proceedings, the 25th International Conference on Machine Learning*, New York, USA, pp. 1096-1103, Jul. 2008.

[41] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *Proc. Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, May 2015, pp. 1–14.

[42] Niko Sünderhauf, Peer Neubert, and Peter Protzel, “Are We There Yet? Challenging SeqSLAM on a 3,000 km Journey Across All Four Seasons,” *Proceedings, Workshop on Long-term Autonomy (ICRA)*, pp. 102-115, May 2013.

[43] Y. Hou, H. Zhang, and S. Zhou, “Convolutional neural network-based image representation for visual loop closure detection,” in *Proc. IEEE Int. Conf. Inf. Autom.*, Lijiang, China, Aug. 2015, pp. 2238–2245.

[44] C. M. Bishop, “Mixture models and EM,” in *Pattern Recognition and Machine Learning*. New York, NY, USA, Springer, 2006, pp. 423–455.

[45] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.* (*CVPR*), San Diego, CA, USA, Jun. 2005, pp. 886–893.

[46] Y. Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell, “Caffe: Convolutional Architecture for Fast Feature Embedding,” *Proceedings, the 22nd ACM International Conference on Multimedia*, Orlando, Florida, USA, pp. 675-678, Nov. 2014.

[47] Tayyab Naseer, Michael Ruhnke, Cyrill Stachniss, Luciano Spinello, and Wolfram Burgard, "Robust Visual SLAM Across Seasons," *Proceedings, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, pp.2529-2535, 28 Sept. to 2 Oct. 2015.

[48] Xin Yang, and Kwang-Ting Cheng, "LDB: An Ultra-Fast Feature for Scalable Augmented Reality on Mobile Devices," *Proceedings, IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, Atlanta, GA, USA, pp.49-57, 5-8 Nov. 2012.

[49] O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Apr. 2015.

[50] J.-J. Huang, W.-C. Siu, and T.-R. Liu, "Fast image interpolation via random forests," *IEEE Trans. Image Process.*, vol. 24, no. 10, pp. 3232–3245, Oct. 2015.

[51] Chu-Tak Li and Wan-Chi Siu, "Fast Monocular Visual Place Recognition for Non-Uniform Vehicle Speed and Varying Lighting Environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1679-1696, Mar. 2020.

[52] Chu-Tak Li, Wan-Chi Siu and Daniel Pak-Kong Lun, "Boosting the Performance of Scene Recognition via Offline Feature-Shifts and Search Window Weights," *Proceedings, IEEE 23rd International Conference on Digital Signal Processing* (*DSP*), Shanghai, China, pp.1-5, 19-21 Nov. 2018.

[53] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba, "Places: A 10 million Image Database for Scene Recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.40, no.6, pp. 1452-1464, Jul. 2017.

[54] Z.-S. Liu, W.-C. Siu, and Y.-L. Chan, "Photo-Realistic Image Super-Resolution via Variational Autoencoders," *IEEE Trans. on Circuits and Syst. for Video Techn.*, 2020.

[55] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, C. C. Loy, Y. Qiao, and X. Tang, "ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks," in *Proc. European Conf. on Comput. Vis. (ECCV)*, 2018.

[56] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-Image Translation with Conditional Adversarial Networks," in *Proc. Computer Vision and Pattern Recognition (CVPR)*, 2017, pp.1125-1134.

[57] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs," in *Proc. Computer Vision and Pattern Recognition (CVPR)*, 2018, pp.8798-8807.

[58] J. Johnson, A. Alahi, and F.-F. Li, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution," in *ECCV*, 2016, pp. 694-711.

[59] L. A. Gatys, A. S. Ecker, and M. Bethge, "A Neural Algorithm of Artistic Style," *arXiv:1508.06576*, 2015.

[60] L.-W. Wang, Z.-S. Liu, W.-C. Siu, and D. P.-K. Lun, "Lightening network for low-light image enhancement," *IEEE Trans. on Image Processing*, 2020.

[61] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image Inpainting for Irregular Holes Using Partial Convolutions," in *Proc. European Conf. on Computer Vision (ECCV)*, 2018, pp. 85-100.

[62] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. Huang, “Free-Form Image Inpainting with Gated Convolution,” in *CVPR*, 2019.

[63] S. Ge, C. Li, S. Zhao, and D. Zeng, “Occluded Face Recognition in the Wild by Identity-Diversity Inpainting,” *IEEE Trans. on Circuits and Syst. for Video Techn.*, 2020.

[64] Z. Yi, Q. Tang, S. Azizi, D. Jang, and Z. Xu, “Contextual Residual Aggregation for Ultra High-Resolution Image Inpainting,” in *CVPR*, 2020.

[65] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros, “Context Encoders: Feature Learning by Inpainting,” in *Proc. Computer Vision and Pattern Recognition (CVPR)*, 2016, pp.2536-2544.

[66] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li, “High-Resolution Image Inpainting using Multi-Scale Neural Patch Synthesis,” in *Proc. Comput. Vis. and Pattern Recog. (CVPR)*, 2017, pp.6721-6729.

[67] S. Iizuka, E. S.-Serra, and H. Ishikawa, “Globally and Locally Consistent Image Completion,” in *ACM Trans. Graph.*, vol. 36, no. 4, Jul. 2017.

[68] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, “Generative Image Inpainting with Contextual Attention,” in *Proc. Computer Vision and Pattern Recognition (CVPR)*, 2018, pp.5505-5514.

[69] Y. Wang, X. Tao, X. Qi, X. Shen, and J. Jia, “Image Inpainting via Generative Multi-column Convolution Neural Networks,” in *Proc. Advances in Neural Inform. Process. Syst. (NeurIPS)*, 2018, pp. 331-340.

[70] K. Nazeri, E. Ng, T. Joseph, F. Z. Qureshi, and M. Ebrahimi, “EdgeConnect: Generative Image Inpainting with Adversarial Edge Learning,” *arXiv:1901.00212*, 2019.

[71] Y.-G. Shin, M.-C. Sagong, Y.-J. Yeo, S.-W. Kim, and S.-J. Ko, “PEPSI++: Fast and Lightweight Network for Image Inpainting,” *IEEE Trans. on Neural Networks and Learning Systems*, 2020.

[72] J. Yang, Z. Qi, and Y. Shi, “Learning to Incorporate Structure Knowledge for Image Inpainting,” in *AAAI*, 2020.

[73] Z. Hui, J. Li, X. Gao, and X. Wang, “Image Fine-grained Inpainting,” *arXiv:2002.02609*, 2020.

[74] S. Y. Kim, K. Aberman, N. Kanazawa, R. Garg, N. Wadhwa, H. Chang, N. Karnad, M. Kim, and O. Liba, “Zoom-to-Inpainting: Image Inpainting with High Frequency Details,” *arXiv:2012.09401*, 2020.

[75] C.-T. Li, W.-C. Siu, Z.-S. Liu, L.-W. Wang, and D. P.-K. Lun, “DeepGIN: Deep Generative Inpainting Network for Extreme Image Inpainting,” in *Proc. European Conference on Computer Vision (ECCV)*, 2020, pp. 5-22.

[76] Y. Zeng, Z. Lin, H. Lu, and V. M. Patel, “Image Inpainting with Contextual Reconstruction Loss,” *arXiv:2011.12836*, 2020.

[77] I. Goodfellow, J. P.-Abadie, M. Mirza, B. Xu, D. W.-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative Adversarial Nets,” in *NeurIPS*, 2014, pp. 2672-2680.

[78] AIM2020: Aim 2020 extreme image inpainting challenge: Methods and results. *In: ECCV Workshops* (2020).

[79] M. Delbracio, H. Talebi, and P. Milanfar, "Projected Distribution Loss for Image Enhancement," *arXiv:2012.09289*, 2020.

[80] Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ade20k dataset. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[81] Zhou, B., Zhao, H., Puig, X., Xiao, T., Fidler, S., Barriuso, A., Torralba, A.: Semantic understanding of scenes through the ade20k dataset. *International Journal on Computer Vision*, 2018.

[82] Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[83] Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[84] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778, 2016.

[85] Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp.7794-7803, 2018.

[86] Liu, Z.S., Wang, L.W., Li, C.T., Siu, W.C., Chan, Y.L.: Image super-resolution via attention based back projection networks. In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pp. 3517-3525, 2019.

[87] Haris, M., Shakhnarovich, G., Ukita, N.: Deep back-projection networks for super-resolution. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1664-1673, 2018.

[88] Shi, W., Caballero, J., Huszar, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., Wang, Z.: Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1874-1883, 2016.

[89] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[90] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. In: *Advances in neural information processing systems*, pp. 8026-8037, 2019.

[91] Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4401-4410, 2019.

[92] Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: *2007 IEEE conference on computer vision and pattern recognition*, pp. 1-8, 2007.

[93] Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, vol. 13, no.4, pp.600-612, 2004.

[94] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: *Advances in neural information processing systems*, pp. 6626-6637, 2017.

[95] Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 586-595, 2018.

[96] S. I. Serengil, and A. Ozpinar, "LightFace: A Hybrid Deep Face Recognition Framework," in *Proc. IEEE Conference on Innovations in Intelligent Systems and Applications* (*ASYU*), 2020, pp.23-27.

[97] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid Scene Parsing Network," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*), 2017, pp.1-11.

[98] Marvin Chancán and Michael Milford, "DeepSeqSLAM: A Trainable CNN+RNN for Joint Global Description and Sequence-based Place Recognition," in *Proc. of the 34th Conference on Neural Information Processing Systems* (*NeurIPS*), 2020.

[99] Stephen Hausler, Sourav Garg, Ming Xu, Michael Milford, and Tobias Fischer, "Patch-NetVLAD: Multi-Scale Fusion of Locally-Global Descriptors for Place Recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*), 2017, pp.14141-14152.

[100] Yuqian Zhou, Connelly Barnes, Eli Shechtman and Sohrab Amirghodsi, "TransFill: Reference-guided Image Inapinting by Merging Multiple Color and Spatial Transformations," *arXiv preprint arXiv:2103.15982*, Mar. 2021.

[101] Yanhong Zeng, Jianlong Fu, Hongyang Chao, and Baining Guo, "Aggregated Contextual Transformations for High-Resolution Image Inpainting," *arXiv preprint arXiv:2104.01431v1*, Apr. 2021.

[102] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park and Victor Lempitsky, "Resolution-robust Large Mask Inapinting with Fourier Convolutons," *arXiv preprint arXiv:2109.07161v2*, Sept. 2021.