January 2020 M.Sc. in ELECTRONICS AND COMPUTER ENGINEERING AMMAR TELLO

HASAN KALYONCU UNIVERSITY

GRADUATE SCHOOL OF
NATURAL & APPLIED SCIENCES

VISUAL PLACE RECOGNITION WITH DTW BASED ENCODED DEEP FEATURES

M. Sc. THESIS
IN
ELECTRONICS AND COMPUTER ENGINEERING

BY

AMMAR TELLO

January 2020

VISUAL PLACE RECOGNITION WITH DTW BASED ENCODED DEEP FEATURES

M.Sc. Thesis

In

Electronics and Computer Engineering

Hasan Kalyoncu University

Supervisor(s)
Dr. Saed ALQARALEH

By
Ammar TELLO
January 2020

GRADUATE SCHOOL OF NATURAL & APPLIED SCIENCES INSTITUTE
MSc ACCEPTANCE AND APPROVAL FORM

Electronics-Computer Engineering M.Sc. (Master Of Science) programme student **Gürcan SERBEST** prepared and submitted the thesis titled **"A Study On Representing Cultural Heritage By Virtual and Augmented Reality"** defended successfully on the date of 13/01/2020 and accepted by the jury as an M.Sc. thesis.

Position	**Title, Name and Surename** **Department/University**	**Signature:**
M.Sc. Supervisor **Jury Head**	Assoc. Prof. Dr. Tolgay KARA Electrical and Electronic Engineering Department Gaziantep University	[illegible]
Jury Member	Assist. Prof. Dr. Sead ALQARALEH Computer Engineering Department Hasan Kalyoncu University	[illegible]
Jury Member	Assist. Prof. Dr. Bülent HAZNEDAR Computer Engineering Department Hasan Kalyoncu University	[illegible]

This thesis is accepted by the jury members selected by the institute management board and approved by the institute management board.

[illegible]

Prof. Dr. Mehmet KARPUZCU

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Ammar TELLO

Signature

[illegible]

ABSTRACT

VISUAL PLACE RECOGNITION WITH DTW BASED ENCODED DEEP FEATURES

TELLO, Ammar
M.Sc. in Electronics and Computer Eng.
Supervisor: Asst.Prof.Dr. Saed ALQARALEH
January 2020, 55 pages

Visual Place Recognition (VPR) techniques have opened the possibilities for autonomous robots and driverless cars to localize itself in a cheap and accurate way using only visual input. Previously, sensors-based system, which uses GPS and distance sensors were frequently used. However, its disadvantages such as the cost and the vulnerability to the signal inference, in addition to the quality improvement in the visual sensor (Camera) lead to replacing such systems with visual-based systems. This system-based is capable of getting input rich with information that is important for a wide range of applications including VPR. As a result, many visualization techniques were examined and multiple categories of image descriptors were injected into some localization algorithms, for the purpose of making a system that is able to be aware of the surrounding environment just like humans.

In this thesis, a new VPR approach is introduced. This approach uses the Dynamic Time Warping (DTW) and features extracted from a Convolutional Neural Network (CNN) architecture that will be encoded by the Fisher Vector (FV). In more detail, the features are extracted from a pre-trained CNN, then, fed into FV to be encoded and finally pushed to the DTW algorithm that will be used to find the best matches between the reference images and the new coming images (test images). In addition, the performance of different CNN architectures was investigated to find the best architecture fit with DTW, and the performance of all layers from all architectures was compared as well. Furthermore, the advantage of replacing the handcrafted features with deep features was also studied.

As the main aim of this work is to develop a robust approach that can face real-life challenges, the deep features are encoded with FV, which we believe can lead to getting more robust features. Our approach was evaluated against other classical approaches, SVM in particular, which was outperformed by our approach especially when it is required to process dataset(s) that has some challenges such as the viewpoint and/or appearance.

Key Words: Dynamic time warping, Deep features, Fisher Vector, CNN, Image sequence matching, Visual place recognition.

ÖZET

VISUAL PLACE RECOGNITION WITH DTW BASED ENCODED DEEP FEATURES

TELLO, Ammar
Yüksek Lisans Tezi, Elektronik-Bilgisayar Müh.
Tez Yöneticisi: Dr. Öğr. Üyesi. Saed ALQARALEH
Ocak 2020, 55 sayfa

Optik Yer Tanımlama (VPR) teknikleri otonom robotların ve sürücüsüz araçların, bir tek görsel girdiler kullanarak ucuz ve doğru bir şekilde yer belirleme imkanı sağlamaktadır. Önceden, Global Konumlandırma Sistemini kullanan algılayıcı tabanlı sistem GPS ve bunun yanı sıra mesafe algılayıcısı sık kullanılmıştır. Ancak, sinyal sonuç çıkarımında maliyet ve korunmasızlık gibi dezavantajlar ve bunula birlikte görsel algılayıcı (Kamera) kalite geliştirmesi gibi sistemlerin, görsel tabanlı sistemlerle değiştirilmesine yol açmaktadır.
Bu sistem tabanlı cihazı bilgi ile zengin bir girdi elde edebilir ve VPR dahil, çok çeşitli uygulamalar için dikkate değer bir öneme sahiptir. Sonuç olarak, birçok görüntüleme tekniği incelenmiş ve farklı görüntü tanımlayıcıları yerelleştirme algoritmalarına yerleştirilmiş ve çevredeki ortamın farkında olacak bir sistem tıpkı insan gibi yapmayı amaçlanmaktadır.
Bu tezde, yeni bir VPR yaklaşımı gösterilmiş ve Dinamik Zaman Çarpıtma (DTW) tekniği kullanılarak Fisher Vector (FV) vasıtasyla kodlanacak olan Evrişimli Sinir Ağı (CNN) yapısından çıkarılan özellikleri kullanılmıştır. Daha ayrıntılı anlatmak gerekirse, özellikler; önceden eğitilmiş bir CNN'den ihraç edilir, daha sonra kodlanması için FV'ye beslenir, sonunda DTW algoritmasına itilir ve referans görüntüler ve yeni gelen görüntüler arasında (test görüntüleri) en iyi eşleşmeleri bulmak için bu şekilde kullanılır. Ayrıca, DTW'ye en fit olanı bulmak için farklı CNN yapılarının performansı araştırıldı ve tüm yapı katmanlarının performansı karşılaştırıldı. Bundan başka, el yapımı özelliklerin, derin özelliklerle değiştirme avantajı da incelenmiştir.
Bu çalışmanın ana hedefi, Hayatın farklı gerçek zorluklarıyla yüzleşebilecek sağlam bir yaklaşım geliştirmek ve FV ile kodlanmış derin özellikleri daha sağlam özellikler

elde edilmesine yol açabileceğine inanıyoruz. Basettiğimiz bu girişm diğer klasik yaklaşımlara karşı ters olduğu değerlendirilmişti, Özellikle SVM olanı bizim yaklaşımımızdan ve bilhassa veri kümeleri işlenmesi gerektiğinde daha iyi performans gösterip yalnız açı ve / veya görünüm gibi bazı zorlukları bulunmaktadır.

Anahtar Kelimeler: Dinamik Zaman Çarpıtma, Derin Özellikler, Fisher Vector, CNN, Görüntü Dizisi Eşlemesi, Görsel Yer Tanımlama.

DEDICATION

This thesis is dedicated to…

My Mother…
Because of you, I am what I am today. Thank you.

My Father…
I cannot thank you enough for what you have done for me. You were always there when I needed you the most.

Layana and Mayyar…
Appreciation is a word that describes how I feel for you being a part of my life

Grandmother, Manal, Maisaa and Yaser…
Only God knows how much I love you.

Batoul…
My beloved, thank you for being there for me.

ACKNOWLEDGEMENTS

I wish to express my deepest gratitude to my supervisor Asst.Prof.Dr. Saed ALQARALEH for his guidance, advice, criticism, encouragements and insight throughout the research.

I would also like to express my sincere gratitude to Asst.Prof.Dr. A.H ABDUL HAFEZ for the continuous support of my M.Sc study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research. Without his guidance and constant feedback, this M.Sc would not have been achievable.

Besides my advisor, I would like to thank the rest of my thesis committee: Assoc.Prof.Dr. Tolgay KARA and Asst.Prof.Dr. Bülent HAZNEDAR for letting my defense be an enjoyable moment, and for your brilliant comments and suggestions, thanks to you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

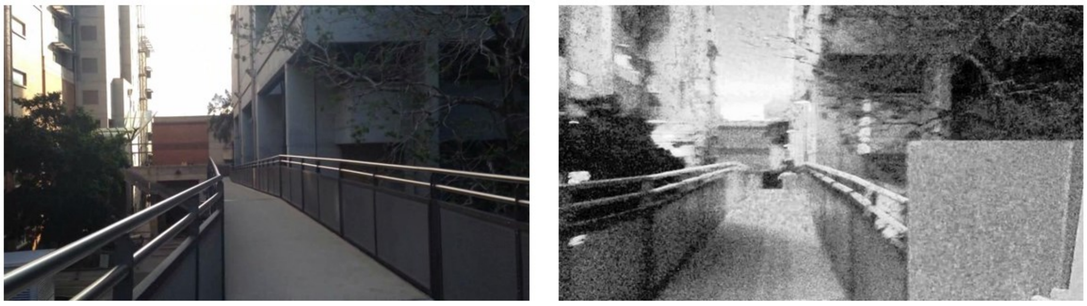
LIST OF SYMBOLS/ABREVIATIONS

ABLE-M	Able for Binary-appearance Loop-closure Evaluation - Monocular
ABLE-P	Able for Binary-appearance Loop-closure Evaluation - Panoramic
ABLE-S	Able for Binary-appearance Loop-closure Evaluation - Stereo
AP	Average Precision
AUC	Area Under Curve
BOW	Bag Of Words
BOVW	Bag Of Visual Words
C	Cumulative distance in DTW
CNN	Convolutional Neural Network
conv	Convolution
DOG	Difference Of Gaussians
DP	Dynamic Programming
DTW	Dynamic Time Warping
EM	Expectation Maximization
FV	Fisher Vector
GMM	Gaussian Mixture Model
HoG	Histogram of Gradients
IFV	Improved Fisher Vector
LDB	Local Difference Binary
ORB	Oriented FAST and Rotated BREIF
PRC	Precision-Recall Curve
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization And Mapping
SURF	Speeded-Up Robust Features
SVM	Support Vector Machine
UAV	Unmanned Aerial Vehicles
UGV	Unmanned Ground Vehicles
UUV	Unmanned Underwater Vehicles
VLAD	Vector of Locally Aggregated Descriptors
VPR	Visual Place Recognition

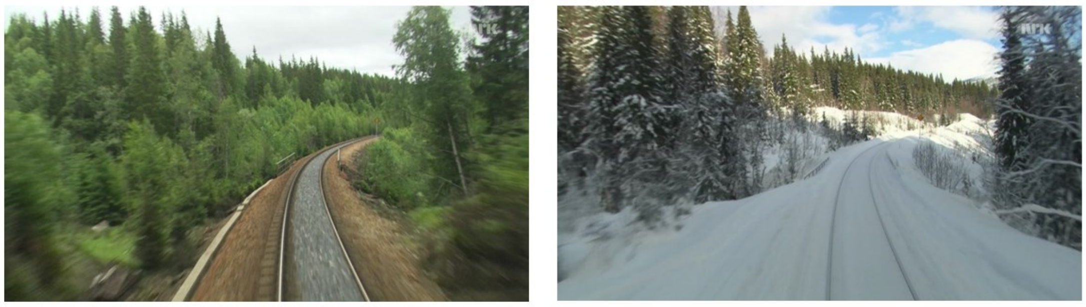
CHAPTER 1

INTRODUCTION

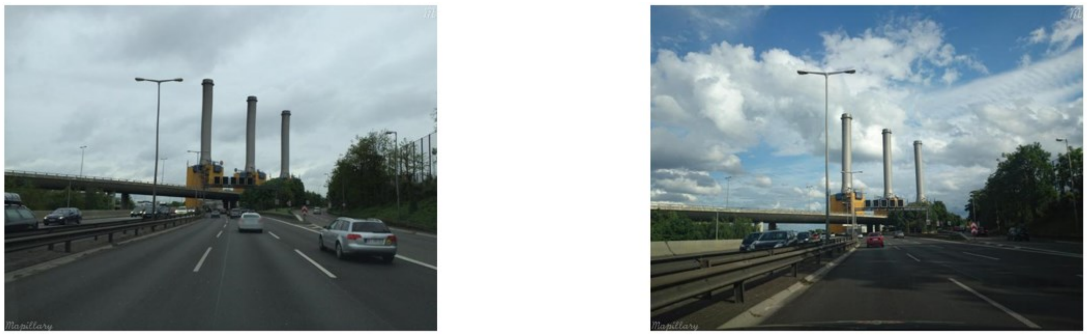
Visual Place Recognition (VPR) refers to how the robot can localize itself using only a visual input of a revisited place. In the last decade, the (VPR) or what called Visual Localization, received significant attention from the research community due to the importance of this task in the robotic field especially for autonomous robots and self-driving cars. It is considered as a challenging problem as appearance can change for the same place over seasons and from day to night and even changes to the place itself, also the variation in viewpoint when the same place revisited again is also a big challenge. Some examples of these challenges are depicted in Figure 1.1. Even though that there are other methods exist for localization task, like Global Positioning System (GPS) based methods, VPR is still preferable due to the significant information that can be retrieved from images and also because of the lack of GPS info in terms of occlusion and absence of the signal.



a



b



c

Figure 1.1 Examples of the challenges faced by the VPR systems. a) The appearance of the same place on day and night, b) Appearance of the same place in summer and winter, c) Appearance of the same place from different viewpoints.

1.1 Problem Statement

Dozens of works in the literature tried to solve the VPR problem as shown in (Lowry et al., 2015). In general, the following components, which are explained in detail in the following chapter, are essential in such systems: 1) Visual Map, 2) Feature Extraction and 3) Localization. Other components like visual perception, motion estimation, and decision which is the output of this system as depicted in Figure 1.2 can be available too. Quite good efforts were made on the different VPR components using general-purpose datasets like ImageNet (Krizhevsky et al., 2012) and special-purpose datasets like Places (Zhou et al., 2017) and SPED (Chen et al., 2017a) which are specified for the VPR tasks. In addition, in these studies many feature extraction approaches including handcrafted and deep features were investigated, these features are integrated with various localization algorithms to achieve the localization task in the best accurate way.

As a result, VPR algorithms like SeqSLAM and FAB-MAP were introduced, which was a big step towards the perfect VPR system. SeqSLAM was great in terms of the way used to deal with the VPR problem which was simplified into two main problems: the first one is to find the best match locally by defining the most similar neighborhood reference images from the visual map, and the second one is to find the best match through the selected reference images. Also, FAB-MAP was the first approach that attempted to solve the VPR problem as a sequence matching instead of image to image matching method.

Despite all these efforts, there still some drawbacks in the existing approaches need to be solved. For example, SeqSLAM has achieved a great performance unless it was vulnerable to the drastic changes in the environment because it was taking the whole image as an input for the localization step. FAB-MAP also has a drawback in the long-term navigation due to the changes in the appearance which make it not suitable for such cases.

So, there still a need to find some approach that is able to perform the localization task in a way that is comparable to what humans are capable of.

Introducing a new approach by improving two main components, i.e., feature extraction and localization modules, that can give promising results in achieving the VPR task, is our aim. For the localization algorithm, a Dynamic Programming (DP) is used represented by Dynamic Time Warping (DTW) which is a sequence alignment algorithm that was used for the first time in visual recognition by (Hafez et al., 2019) to find the best matches among the distance matrix between a reference and test datasets. Then, we integrated the DTW with the well-known deep features, after that, these features were encoded by the Fisher Vector (FV) algorithm.

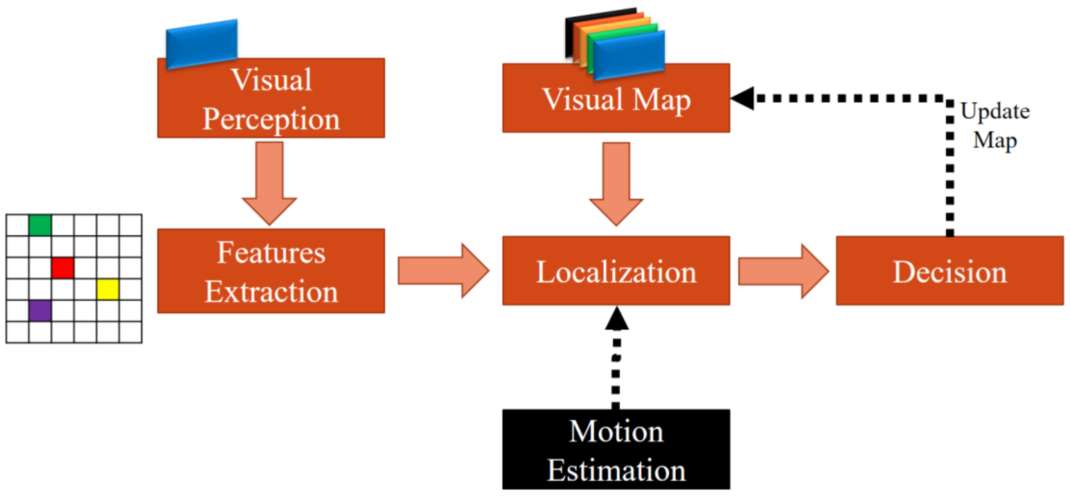


Figure 1.2 Visual Place Recognition Schematic Diagram.

1.2 Motivation and Objectives

The main motivation of this work is to make a breakthrough in the way of treating the VPR problem by introducing a new approach based on DTW that has achieved promising results in (Hafez et al., 2019), which we have further improved by integrating the encoded deep features. In other words, this combination made to get the benefits from both, the DTW and the encoded Deep Features to come with a VPR approach capable of manipulating with different VPR challenges. The main goals that this thesis was attempted to achieve are described as follows:

- Introducing a new VPR approach that can deal with different VPR challenges.
- Investigate the best Convolutional Neural Network (CNN) architecture that gives the best performance when integrated with DTW and finds the best layer in this architecture.
- Find the benefit of using Deep Features instead of handcrafted features.
- Investigate the effect of encoding the deep features with FV in terms of performance.
- Finally, evaluate the performance of the proposed approach against the Support Vector Machine (SVM) classifier which is a classifier used widely as the last layer in CNN architectures.

1.3 Contributions

The main contribution of this work is two folds:

i) Introducing a new VPR algorithm that utilizes DTW and deep features encoded using fisher vector. In more detail, the proposed algorithm formulates the DTW as a visual place recognition system. The feature maps are extracted from a selected convolution layer after applying the test and the reference image sequences as an input to the network and these features are encoded using FV based. Then, the DTW algorithm aligns the two sequences resulted from encoding features by matching each test image to the closest image in the reference sequence of images.

ii) The different layers from the VGG16, ResNet50 and HybridNet networks are explored to identify the layers that are best performing with the DTW algorithm for visual place recognition.

Figure 1.3 shows the output images produced by the DTW algorithm corresponding to an example input image while using different kinds of features.

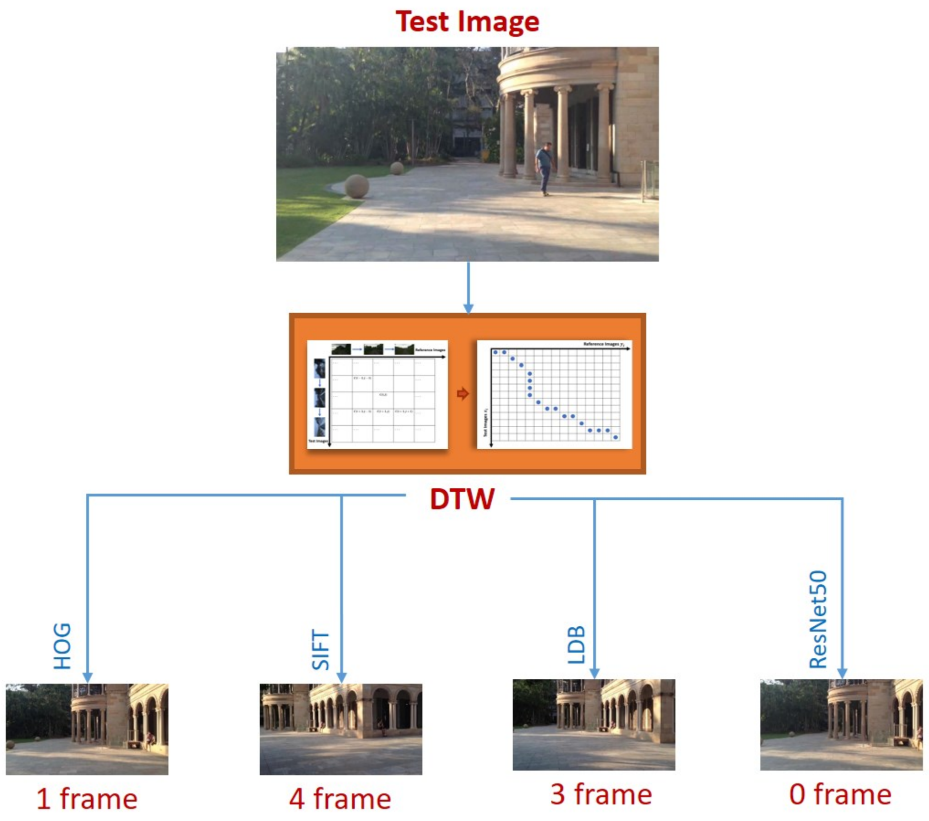


Figure 1.3 Examples of the output of DTW using different features where the difference between the test image and the ground truth image are shown in red.

1.4 Organization of the Thesis and Publications

The rest of this work is organized as follows: The background chapter (CHAPTER 2) explains the main techniques used in this thesis. LITERATURE REVIEW (CHAPTER 3) contains a review of the milestone works in the VPR field. DTW BASED ENCODED DEEP FEATURES (CHAPTER 4) contains details about the proposed approach, EXPERIMENTAL EVALUATION AND ANALYSIS (CHAPTER 5) presents the results of the experiments done through this thesis to achieve the goals described in the section (1.2) of this chapter. CONCLUSION (CHAPTER 6) is to give a brief about the outcomes of this work and some headlines for possible directions for future work.

It is worth mentioning that a conference paper (Hafez et al., 2019) titled as “Visual Place Recognition by DTW-based sequence alignment” was published on “SIU 2019” SIVAS, Turkey. Also, another paper was submitted to the IEEE conference “ICRA 2020” and it is under evaluation until the time of writing this thesis. One more paper is expected to be published into the “SIU 2020” conference as a result of this work.

CHAPTER 2

BACKGROUND

2.1 Visual Place Recognition Components

As mentioned in Chapter 1, the VPR systems share the following main components: 1) Visual Map: which is represented by the images of the visited place or generally the environment, and these images are considered as references, while the new coming images are called test images. 2) Feature Extraction: In this step, each image is represented by a descriptor that is formulated through some feature extraction algorithm(s), which works on finding the most important representatives inside the image. 3) Localization: This component is responsible for finding the best matches between reference and test images, so, the robot can localize itself according to the place that the matched reference image referred to. In the next two sections, more details about Feature Extraction and Localization are given.

2.2 Features Extraction

In general, there are two types of features that can be extracted from images, handcrafted features and deep features.

2.2.1 Handcrafted Features

These features represent images using the information present in the image itself. SIFT, HoG, and LDB (Lowe, 1999; Dalal and Triggs, 2005; Yang and Cheng, 2013), are examples of efficient and frequently used handcrafted descriptors.

2.2.1.1 Scale-Invariant Feature Transforms (SIFT)

SIFT algorithm is a way of detecting and describing the local features in images with scale-invariant, i.e., the same features at different scales are detected as similar features. This is done through different steps started with detecting the features (Key points) using the second derivative of Gaussian or what called Laplacian of Gaussian (LoG) where different standard deviation (σ) values are used to detect the features on different scales. The calculation of LoG is simplified using Difference of Gaussian (DoG) as follows:

$$\frac{\partial G}{\partial \sigma} = \sigma \Delta^2 G \tag{2.1}$$

$$\sigma \Delta^2 G = \frac{\partial G}{\partial \sigma} = \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma} \tag{2.2}$$

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \Delta^2 G \tag{2.3}$$

Where G is the Gaussian density function. So, to calculate the LoG, the Gaussian with different σ values are applied on the image, then, each two consequence results are subtracted to get the DoG, this procedure should be repeated for different scales, where the image is down-sampled in a pyramid way, and for each scale of image, different σ values are applied. In other words, DoG is calculated at different scales of images and different σ values. Then, the key points are extracted, where each pixel is compared with its 26 neighbors, 8 in the current and 18 in the two adjacent scales. If this pixel is larger or smaller than all its neighbors, then it can be selected as a key point. However, some of the selected key points are outliers and must be eliminated. This is done through two methods, the first one is based on Taylor series for DoG:

$$DoG(X) = DoG + \frac{\partial DoG^T}{\partial X} X + \frac{1}{2} X^T \frac{\partial^2 DoG}{\partial X^2} X, \text{ Where } \quad X = (x, y, \sigma)^T \tag{2.4}$$

So, for each (X), if the $|DoG(X)| > threshold$, then, this value can be considered as an accepted key point, otherwise, it is considered as an outlier. The second method is based on Hessian matrix:

$$H = \begin{bmatrix} DoG_{xx} & DoG_{xy} \\ DoG_{xy} & DoG_{yy} \end{bmatrix} \tag{2.5}$$

Where the elements are the derivatives of DoG according to the shown subscript. The trace and determinant of this matrix are given as follows:

$$Tr(H) = DoG_{xx} + DoG_{yy} = \lambda_1 + \lambda_2 \tag{2.6}$$

$$Det(H) = DoG_{xx}DoG_{yy} - (DoG_{xy})^2 = \lambda_1\lambda_2 \quad \textbf{(2.7)}$$

Where λ_1 and λ_2 are the eigenvalues of the H. The outliers are rejected when r>10, where: $r = \frac{\lambda_1}{\lambda_2}$

Then, after eliminating the outliers, a descriptor for each key point should be computed using the orientation and magnitude of a 16x16 patch around each keypoint. A weighted histogram can be formulated by dividing the patch into 4x4 regions, and for each region, a histogram weighted by magnitude and consists of 8 bins is calculated. A 128-length descriptor is formulated for each key point, and the whole image can be described by all descriptors related to the detected keypoints.

2.2.1.2 Histogram of Gradients (HoG)

HoG is a global descriptor that divides the image into several blocks where each block is represented by several cells and each cell consists of several pixels. For each cell, a histogram of gradient is computed, where the gradient orientation is quantized into a specific number of bins and weighted by the gradient magnitude just like in SIFT. The gradient of an image along X and Y axis can be calculated using Sobel filter which can be described as follows:

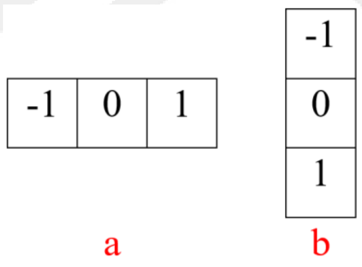


Figure 2.1 Sobel Filter, a) For X direction, b) For Y direction

The magnitude and orientation can be calculated using equations **(2.8)** and (2.9) respectively.

$$g = \sqrt{g_x^2 + g_y^2} \quad \textbf{(2.8)}$$

$$\theta = \arctan\left(\frac{g_y}{g_x}\right) \quad \textbf{(2.9)}$$

The descriptors for all cells are concatenated to formulate a descriptor for the image. The blocks can share some cells according to the stride parameter which specify the number of pixels that each block is shifted from its neighbor.

2.2.1.3 Local Difference Binary (LDB)

LDB is a local binary descriptor, where the key points are detected, then described by LDB. The detecting process could be done based any keypoint detector like FAST and ORB algorithms. For each keypoint, a patch that surrounds this keypoint is defined, and each patch is divided into n*n grids. For each pair of grids, the following function is performed:

$$\tau\big(Func(i), Func(j)\big) := \begin{cases} 1 & if\big(Func(i) - Func(j)\big) > 0 \; and \; i \neq j \\ 0 & otherwise \end{cases} \quad \textbf{(2.10)}$$

Where i and j are different grids, and $Func(.)$ is a function applied on the grid. This function is one of three different functions as follows:

$$Func_{intensity}(i) := \frac{1}{m}\sum\nolimits_{k=1\sim m} Intensity(k) \quad \textbf{(2.11)}$$

$$Func_{dx}(i) := Gradient_x(i) \quad \textbf{(2.12)}$$

$$Func_{dy}(i) := Gradient_y(i) \quad \textbf{(2.13)}$$

Where $Func_{intensity}(.)$ is the average intensity in the grid, m is the number of pixels in that grid. $Func_{dx}(.)$ is the gradient along x direction, and $Func_{dy}(i)$ is the gradient along y direction. So, for each two grids, three binary bits are calculated. The size of each grid is very important in terms of robustness and distinctiveness where smaller size means more distinctiveness and larger size gives more robustness, so, to achieve both, multiple grid sizes are applied. The full LDB descriptor is formulated through concatenating all calculated binary strings from all grids. It is important to note that bit selection procedure is applied to achieve higher efficiency in terms of the time of matching descriptors and the needed storage to maintain the final descriptors. This process is done through one of two approaches: 1) Random selection: where n bits are selected from the full LDB descriptor randomly, 2) Entropy-based selection: where LDB descriptors for a training dataset is generated to make a matrix of size (N x number of records) where N is the length of the LDB descriptor for one image. For each matrix column, the entropy is calculated as follows:

$$entropy_i = -P_i(1)\log\big(P_i(1)\big) - P_i(0)\log\big(P_i(0)\big) \qquad (1 \leq i \leq N) \quad \textbf{(2.14)}$$

$P_i(1)$ is the probability of having 1 in column i and $P_i(0)$ the probability of having 0 in the same column. The final LDB descriptor is generated from n columns with the highest entropies.

2.2.2 Deep Features

Using this method, the image is represented using the output of a specific Layer from a CNN model. In this thesis, we considered the layers of the VGG16, ResNet50 and HybridNet networks (Zhang et al., 2015; He et al., 2016; Chen et al., 2017a). The details of the Deep feature are presented in the next section.

2.2.2.1 Features from Pre-trained CNN Networks

In general, CNN composed of a number of layers such as convolution layer, which mainly aims to detect local conjunctions of features from the previous layer and mapping their appearance to a feature map, pooling layer that responsible for reducing the size of the activation maps, and ReLU layer, which can be considered as special implementation aims to combine non-linearity and rectification layers. Nowadays, multiple CNN models are available and can be integrated and considered as good choices for improving any image retrieval systems. The following are some successful deep Conv Net examples. AlexNet (Krizhevsky et al., 2012) was one of the first deep networks that defeat classification traditional methodologies. In general, it consists of 5 convolutional layers followed by 3 fully connected (FC) layers. VGGNet (Zhang et al., 2015) consists of 16 convolutional layers and can be considered as one of the most preferred choices for extracting features from images. In addition, the weight configuration of the VGGNet is publicly available and has been used in many other applications. GoogleNet (Szegedy et al., 2015) consists of 22 layers, however, it has significantly reduced the number of used parameters by using several very small convolutions, which leads to reducing the number of parameters to 4 million. Overall, it has achieved performance close to the human-level. More recently, ResNet (He et al., 2016) was proposed with a novel architecture based on skip connections and features. Heavy batch normalization was introduced. Hence, the system can be trained using 152 layers while still having lower complexity than other models. Another model that was developed for VPR applications is known as HybridNet (Chen et al., 2017a). It consists of 6 convolutional layers followed by two FC layers. This CNN architecture initialized with weights taken from CaffeNet (Krizhevsky et al., 2012) as both models have the same dimensions for the first five layers, then, the HybridNet was fine-tuned on the SPED dataset which was also proposed in the same work. Details about VGG16, ResNet50, and HybridNet are shown in Table 2.1, Table 2.2 and Table 2.3 respectively. Also, Figure 2.2 shows the visualization of VGG16.

Table 2.1 VGG16 architecture layers including the output size of each layer, the kernel size and the number of feature maps in each layer.

Layer name	Output size	Kernel size, Number of Feature maps
conv1_x	112×112	$\begin{bmatrix} 3\times 3, 64 \\ 3\times 3, 64 \end{bmatrix}$
conv2_x	56×56	$\begin{bmatrix} 3\times 3, 128 \\ 3\times 3, 128 \end{bmatrix}$
conv3_x	28×28	$\begin{bmatrix} 3\times 3, 256 \\ 3\times 3, 256 \\ 3\times 3, 256 \end{bmatrix}$
conv4_x	14×14	$\begin{bmatrix} 3\times 3, 512 \\ 3\times 3, 512 \\ 3\times 3, 512 \end{bmatrix}$
conv5_x	7×7	$\begin{bmatrix} 3\times 3, 512 \\ 3\times 3, 512 \\ 3\times 3, 512 \end{bmatrix}$
FC 6	1×1	4096
FC 7	1×1	4096
Softmax	1×1	1000

Table 2.2 ResNet50 architecture layers including the output size of each layer, the kernel size and the number of feature maps in each layer.

Layer name	Output size	Kernel size, Number of Feature maps
conv1	112×112	7×7, 64, stride 2
conv2_x	56×56	3×3 max pool, stride 2
		$\begin{bmatrix} 1\times 1, 64 \\ 3\times 3, 64 \\ 1\times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 1\times 1, 128 \\ 3\times 3, 128 \\ 1\times 1, 512 \end{bmatrix} \times 4$
conv4_x	14×14	$\begin{bmatrix} 1\times 1, 256 \\ 3\times 3, 256 \\ 1\times 1, 1024 \end{bmatrix} \times 6$
conv5_x	7×7	$\begin{bmatrix} 1\times 1, 512 \\ 3\times 3, 512 \\ 1\times 1, 2048 \end{bmatrix} \times 3$
Softmax	1×1	1000

Table 2.3 HybridNet architecture layers including the output size of each layer, the kernel size and the number of feature maps in each layer.

Layer name	**Output size**	**Kernel size, Number of Feature maps**
conv1	55×55	$11 \times 11, 96$
conv2	27×27	$5 \times 5, 256$
conv3	13×13	$3 \times 3, 384$
conv4	13×13	$3 \times 3, 384$
conv5	13×13	$3 \times 3, 256$
conv6	6×6	$3 \times 3, 256$
FC	1×1	4096
Softmax	1×1	4096

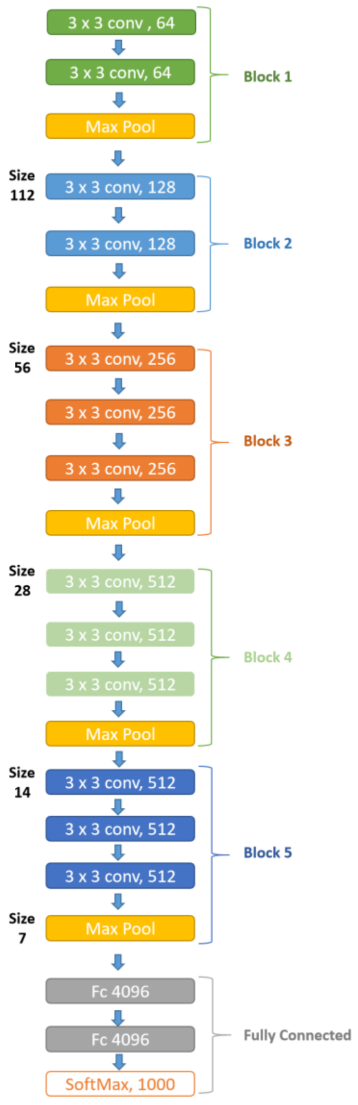


Figure 2.2 A representation of VGG16 layers.

2.3 Encoding Features

BOW, VLAD and FV are examples of encoding methods that enhance the features extracted through feature extraction algorithms. BOW, VLAD and FV are explained below.

2.3.1 Bag of Words (BOW)

BOW or BOVW (Bag of Visual Words) is the first encoding method that was used at the beginning for text retrieval, but then, it became used for image retrieval also. In this work we are interested in the encoding use of this algorithm.

The first step is to detect and describe the features in the images using algorithms like SIFT, SURF, etc. The second step is to build what called Visual Codebook, this can be done by clustering the features extracted from the training dataset based some clustering algorithm like K-Means and its improvements. The centers of the generated clusters are representing the visual words, and now the visual codebook which consists of these visual words became ready to use in the testing phase. Finally, when a new image come, the distance between the extracted features from this image and the visual words should be calculated, each feature vector is assigned to the nearest center, and a histogram can be formulated which represents the number of occurrences of each visual word in the testing image, and according to it, the image can be classified into the category that the most occur visual word is refer to. Also, the generated histogram can be used as a descriptor that represents the image.

2.3.2 Vector of Locally Aggregated Descriptors (VLAD)

VLAD is another encoding method, the first and second steps are the same as BOW where the features should be extracted and clustered to make a visual codebook. To make a VLAD vector for image (I) the residuals should be computed as follows:

$$v_c = \sum_{i=1}^{K} q_{ic} \, (x_i - \mu_c) \qquad \textbf{(2.15)}$$

Where c is the number of cluster, q is the strength of the association with cluster, which has two constraints: $q_{ic} > 0$ and $\sum_{c=1}^{M} q_{ic} = 1$ where M is the number of clusters.

All residuals will be concatenated to formulate the VLAD vector of image (I):

$$\phi(I) = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ v_c \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix} \qquad \textbf{(2.16)}$$

2.3.3 Fisher Vector (FV)

Fisher vector can be described as a way of producing a global descriptor for an image-based on the local descriptors of its local regions, and it is the best among other encoding techniques. Its main steps are explained below.

Step1: Gaussian Mixture Model (GMM)

In general, the Gaussian Mixture Model (GMM) can be used as the step before using the selected encoding schemes, i.e., it is a soft clustering process for the extracted features. In more detail, as mentioned before, the features should be extracted from the image and described using one of the feature extraction methods, then, these features will be soft-assigned to clusters made by GMM. GMM can be described as a set of multivariate Gaussian distributions, represented as a sum of weighted Gaussian distribution functions based equation (2.17):

$$p(x|\lambda) = \sum_{c=1}^{M} w_c \, g(x \,|\mu_c, \Sigma_c) \qquad \textbf{(2.17)}$$

Where w_c is the weight of the c component, $g(x \,|\mu_c, \Sigma_c)$ is the Gaussian density function, x is the data vector, μ_c is the mean of the c Gaussian component and Σ_c is the covariance matrix of the same component. And the Gaussian density function is given by equation (2.18):

$$g(x \,|\mu_c, \Sigma_c) = \frac{1}{(2\pi)^{D/2} \, |\Sigma_c|^{1/2}} \exp\{-\frac{1}{2}(x-\mu_c)' \sum_i^{-1} (x-\mu_c)\} \qquad \textbf{(2.18)}$$

The mixture weights should meet the constraint

$$\sum_{c=1}^{M} w_c = 1 \qquad \textbf{(2.19)}$$

As a result, The GMM can be summarized as follows:

$$\lambda = \{w_c, \mu_c, \Sigma_c\} \quad c = 1, \dots, M \qquad \textbf{(2.20)}$$

Where M is the number of GMM components. Estimating the values of the GMM parameters (w_c, μ_c, Σ_c) is done through an iterative algorithm called Expectation Maximization, which has a concept of maximizing the likelihood of the GMM model, knowing that for each iteration, the GMM model is replaced with a new one that met the following constraint:

$$p(X|\bar{\lambda}) \geq p(X|\lambda) \qquad \textbf{(2.21)}$$

Where $\bar{\lambda}$ is the new model and $p(X|\lambda)$ is the likelihood of the λ model given in equation (2.22):

$$p(X|\lambda) = \prod_{t=1}^{T} p(x_t|\lambda) \qquad \textbf{(2.22)}$$

Here, T is the number of training vectors $X = \{x_1, \dots, x_T\}$.

For each EM iteration, the weights, means and covariance matrices are updated according to equations (2.23), (2.24) and (2.25) respectively

$$\bar{w}_c = \frac{1}{T} \sum_{t=1}^{T} Pr(c|\, x_t, \lambda) \qquad \textbf{(2.23)}$$

$$\bar{\mu}_c = \frac{\sum_{t=1}^{T} Pr(c|\, x_t, \lambda)\ x_t}{\sum_{t=1}^{T} Pr(c|\, x_t, \lambda)} \qquad \textbf{(2.24)}$$

$$\bar{\sigma}_c^2 = \frac{\sum_{t=1}^{T} Pr(c|\, x_t, \lambda)\ x_t^2}{\sum_{t=1}^{T} Pr(c|\, x_t, \lambda)} - \bar{\mu}_c^2 \qquad \textbf{(2.25)}$$

Where $Pr(c|\, x_t, \lambda)$ is the Posteriori of the c component explained in equation **(2.26)**, x_t is the data vector and λ is the current GMM model. The outcome of the EM algorithm is a GMM model that best fit with the given training data.

$$Pr(c|\, x_t, \lambda) = \frac{w_c\, g(x_t\, |\mu_c, \Sigma_c)}{\sum_{k=1}^{M} w_k\, g(x_t\, |\mu_k, \Sigma_k)} \qquad \textbf{(2.26)}$$

Step 2: Obtaining the Fisher Vector

The fisher vector is built on fisher kernel which is responsible of finding the similarity between two data input using a generative model, in this case it is a GMM model. As a result, each component of each data vector has two elements computed as follows:

$$u_{dc} = \frac{1}{T\sqrt{\pi_c}} \sum_{t=1}^{T} Pr(c|\, x_t, \lambda)\ \frac{x_{dt} - \mu_{dc}}{\sigma_{dc}} \quad \textbf{(2.27)}$$

$$v_{dc} = \frac{1}{T\sqrt{2\pi_c}} \sum_{t=1}^{T} Pr(c|\, x_t, \lambda) \left[\left(\frac{x_{dt} - \mu_{dc}}{\sigma_{dc}}\right)^2 - 1\right] \quad \textbf{(2.28)}$$

Where $d = \{1, \dots, K\}$ is a component from the data vector x with dimension K.

The final fisher vector of image (I) is represented as follows:

$$\phi(I) = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ u_c \\ \cdot \\ \cdot \\ v_c \\ \cdot \\ \cdot \end{bmatrix} \quad \textbf{(2.29)}$$

The dimension of this fisher vector can be given in equation (2.30):

$$Dimension\ of\ FV = M \times K \times 2 \quad \textbf{(2.30)}$$

In this work, we are interested in the improved version of the Fisher Vector (Improved Fisher Vector (IFV)) (Perronnin et al., 2012), since it has been proved that it can get better performance compared to the classical FV.

2.3.4 Improved Fisher Vector (IFV)

Two more steps were added to the FV algorithm that leads to improve the performance. The first step is to normalize the FV for each dimension using the following function

$$f(\phi) = sign(\phi)|\phi|^{\alpha} \quad \textbf{(2.31)}$$

Where α is a parameter in the range [0, 1].

The second step is to normalize the resulted FV from the previous step with L2 normalization **(2.32)**.

$$\|\phi\|_2 = \frac{\phi}{\sqrt{\phi_1^2 + \phi_1^2 + \cdots + \phi_{Dimension\ of\ FV}^2}} \tag{2.32}$$

2.4 Localization

The localization is a component that is responsible for deciding whether the current place is a new place or it's a revisited place giving the corresponding image from the reference images (visual map). Many algorithms such as the DTW and SVM can be used for this purpose.

2.4.1 DTW

In general, the DTW can be used to align two sequences of images, in our case, it is the test and the priori annotated reference sequences, which is represented by the features vectors Ax_i and Ay_j, where x_i is an image in the test sequence and y_j is an image in the reference sequence. Then, the distance between each two features vectors extracted from reference and test sequences is explained in equation (2.33)

$$D(i,j) = 1 - Distance(x_i, y_j) = \frac{|Ax_i| \,.\, |Ay_j|}{\|Ax_i\| \cdot \|Ay_j\|} \tag{2.33}$$

Briefly, the distance value between the image x_i from the test sequence and the image y_j from reference sequence is stored in $D(i,j)$. It is worth mentioning that the proposed algorithm can work with any distance matrix such as cosine and Euclidean and independent of the selected cost function, the DTW will always follow the same set of steps. This distance matrix is depicted in Figure 2.3. However, in our work, the Cosine similarity was used, which make it necessary to make some changes to the way that DTW is follow to find the best path, details about these changes can be found on chapter 4 (DTW BASED ENCODED DEEP FEATURES).

After that, the Equation (2.34) is used to fill the cost matrix by accumulating the elements of the distance matrix (D). In other words, in the developed algorithm, the (2.34) relation is used to fill the cost matrix, which represents the sum of the distance between current matching two images and the minimum of the cumulative distances of the neighboring images.

$$C(i,j) = D(i,j) + min \begin{cases} D(i-1,j), \\ D(i,j-1), \\ D(i-1,j-1), \end{cases} \quad \textbf{(2.34)}$$

Whenever the matrix C is filled out, DTW works on defining an optimal path of matches P, which is the result of backward tracing in the matrix C choosing the previous elements with the lowest cumulative distance. Hence, the path through the elements of matrix C that has the minimum sum of cost values $C(i,j)$ is the optimal path and can be interpreted as minimizing the following function.

$$Q(P) = \sum_{l=1}^{L} C(i_l, j_l) \quad \textbf{(2.35)}$$

An example of this path is shown in Figure 2.4. For more details about DTW algorithm, the reader is referred to (Sakoe and Chiba, 1978; Kate et al., 2016; Petitjean et al., 2016).

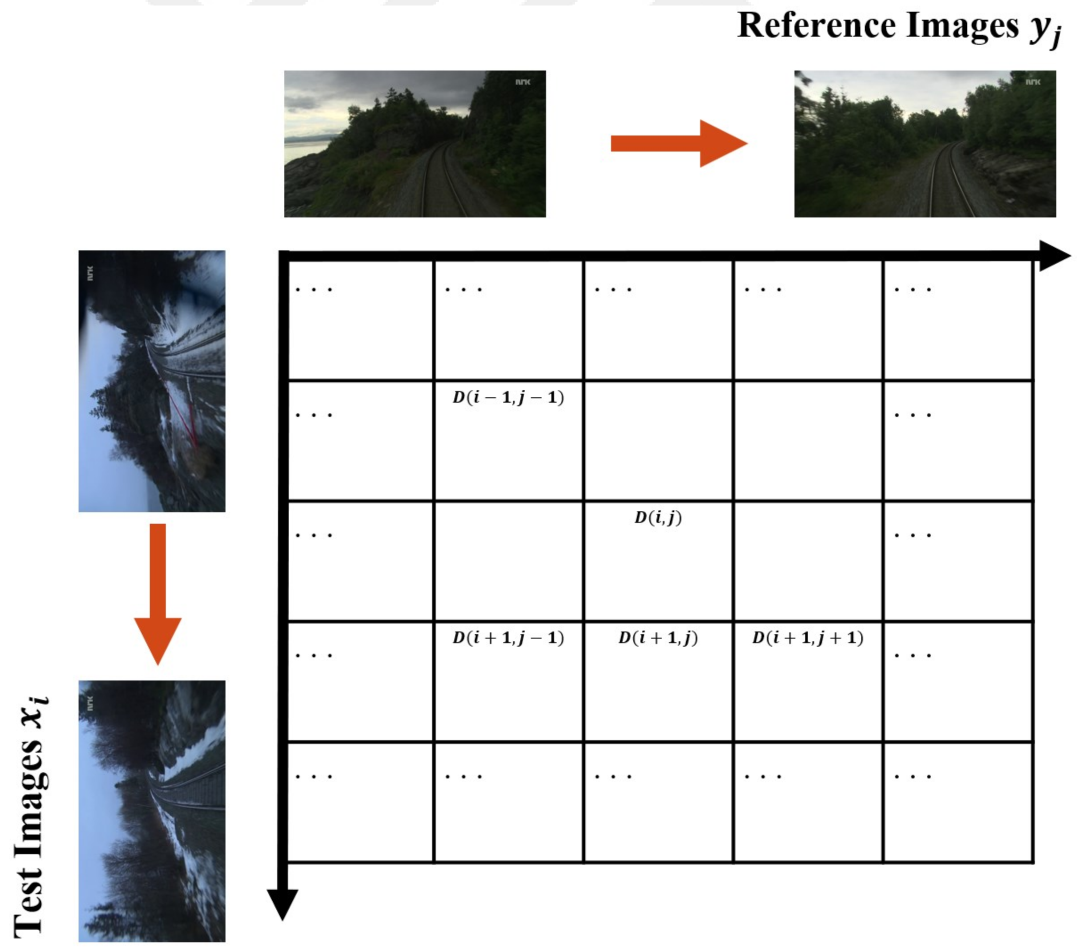


Figure 2.3 The distance matrix between the features vectors of the reference images and test images.

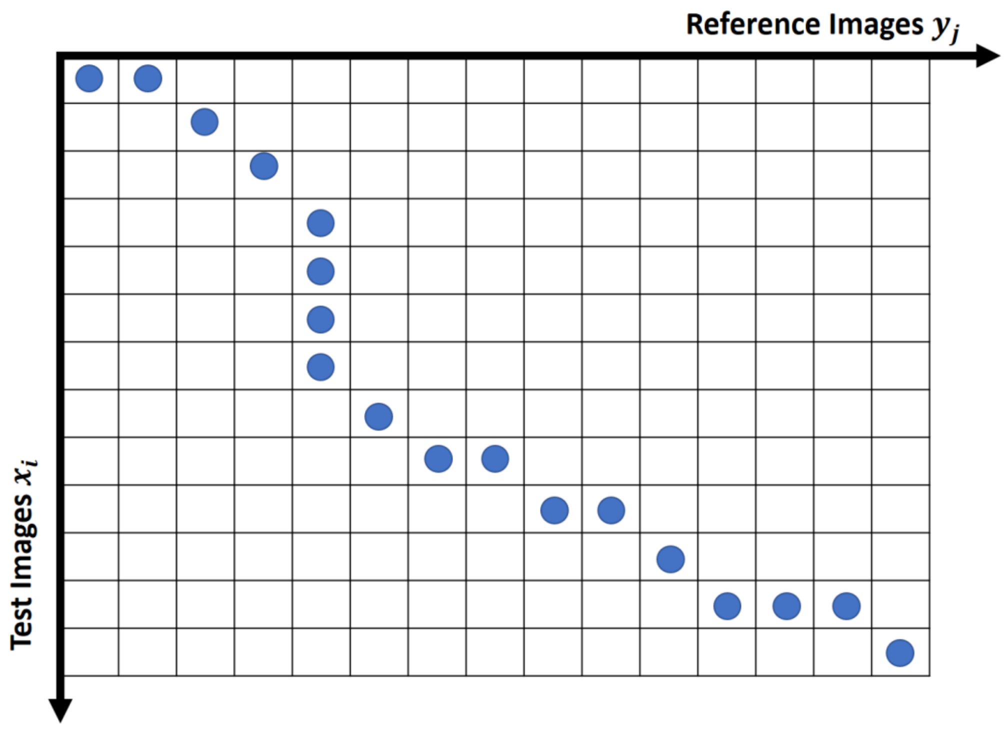


Figure 2.4 The best path between the reference images and test images using DTW.

CHAPTER 3

LITERATURE REVIEW

Visual Place Recognition (VPR) has been well studied in the literature because of the importance of this field in terms of achieving the localization task for autonomous robots and vehicles based visual input (Cummins and Newman, 2008). The recent researches about autonomous robots based on computer vision techniques and different categories of VPR approaches (Lowry et al., 2015) are summarized in the following sections.

3.1 Autonomous Robots Based on Computer Vision

Localization and Navigation are the key skills that autonomous robots should have to be able to recognize its current location and how to reach the desired destination(s). Different kinds of unmanned robots are existed, including Unmanned Ground Vehicles (UGVs) (Bojarski et al., 2016), Unmanned Aerial Vehicles (UAVs) (Milford et al., 2011) and Unmanned Underwater Vehicles (UVVs) (Hong et al., 2016). Examples of those kinds are shown in Figure 3.1.

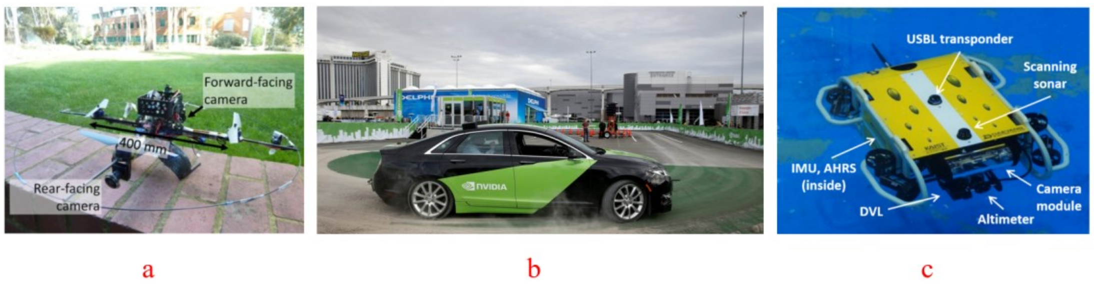


Figure 3.1 Examples of autonomous robots. a) UAV, b) UGV, c) UUV.

In this work, we focused on the VPR approaches in terms of UGVs like driverless cars. The most recent works in this field are focusing on using what called Visual Simultaneous Localization and Mapping (SLAM). In such methods, the sensors responsible for the localization task like GPS are replaced or merged with a visual sensor that can make the localization and mapping tasks at the same time. One of the most recent works in this field called ORB-SLAM (Mur-Artal et al., 2015). This algorithm is initiated by describing the new coming frame based the Oriented FAST and Rotated BREIF (ORB) binary descriptor. In addition, this system contains a Place Recognition module based on the BOW to perform the re-localization procedure when the tracking module is lost.

3.2 Approaches based on the Handcrafted Features

This category can use local descriptors and/or global descriptors. For local descriptors, key points in the image should be detected, then, the region around each key point is described. For global descriptors, there is no need to detect the key points. Instead, a descriptor for the whole image is generated.

3.2.1 Local Descriptors

Local descriptors like scale-invariant feature transform (SIFT) (Lowe, 1999) and speeded-up robust features (SURF) (Bay et al., 2006) were used in the VPR approaches. In addition, the Fast Appearance-Based Mapping (FAB-MAP) algorithm (Cummins and Newman, 2008) is a sequence matching approach that is based on this category. In more detail, this approach performs the sequence matching between two sets of images using SURF features as descriptors encoded using Bag of visual words (BOW) algorithm. This is done by clustering the descriptors of the local regions generated by (SURF) into a predefined number of clusters, which is considered as a visual codebook obtained from the training dataset. Related to the testing dataset, it will be described through (SURF), where each local descriptor is assigned into the nearest cluster using KD-Tree algorithm. Moreover, the resulted visual words are used to build the mutual information graph, where the node refers to a visual word, and the weight on the edge connecting two nodes represents the mutual information. The spanning tree of this graph is what called Chow Liu tree which is a representation of the visual word distribution that works as a generative model, so, the coming images can be considered as a pre-visited location or a new location by knowing how likely it belongs to the available locations distributions.

3.2.2 Global Descriptors

The second category, which uses descriptors such as Histogram of Gradients (HoG) (Dalal and Triggs, 2005) represents the whole scene with a global descriptor. SeqSLAM (Milford and Wyeth, 2012) is another sequence matching approach that manipulates images by getting a global descriptor for the image, and then it calculates the distance of the image descriptor to all reference images to find the best match. Although, this system has achieved a good performance, since it is based on the image itself makes it vulnerable to any changes in the environment.

In (Naseer et al., 2014), flow control is represented as a directed graph where a source and sink nodes are available and the best path between these two nodes should be found to achieve a sequence matching between two sets of images.

Another method uses the ABLE (Arroyo et al., 2014) algorithm which is based on the binary descriptor, i.e., Local Difference Binary (LDB). It has been found that this method is good when illumination cases the variance between the two sets of images, as this method has a specific parameter that can eliminate the variation between the two sequences. In addition, it processes a sequence of images instead of processing one image, which makes it suitable for the long-term scenarios. Multiple versions of this algorithm were introduced to deal with multiple input scenarios including monocular (ABLE-M) Stereo (ABLE-S) and panoramic (ABLE-P).

3.3 Approaches based Deep Features

Different approaches were proposed to solve the VPR problem based on deep features. In general, each of these approaches has adopted one of the follows features extraction methods: 1) The pixels of the whole image are the input of the CNN and then features are extracted from some layer to represent the image (Chen et al., 2014). 2). Secondly, some landmarks could be defined by human and the features of these regions are extracted from the output of convolutional layers (Sünderhauf et al., 2015). 3) The third option is the same as the first one except that instead of getting the output of the CNN directly, landmarks are detected automatically and formulated to produce a representation for the whole image (Chen et al., 2017b; Khaliq et al., 2018, 2019).

In the last decade, deep learning and especially pre-trained CNN is frequently used, which is a network architecture trained on a dataset collected for some computer vision application, and then, the trained weights can be used to get the output or extract features from the deep layers using another dataset for achieving the same or different purpose of the original training dataset. In the following sub-section, some recent studies related to the use of pre-trained CNN are summarized.

3.3.1 Using Pre-Trained Models

It has been proved that tasks like image retrieval and image classification could be achieved using pre-trained CNN (Wang et al., 2017a, b). The deep features can be extracted from any layer of the used CNN. In (Sünderhauf et al., 2015b; Chen et al., 2018; Yue-Hei Ng et al., 2015) and (Chandrasekhar et al., 2017) the performance of tasks like image retrieval, image classification and place recognition, based features extracted from different layers of a CNN, was investigated. The results of these studies showed that the last layer is considered as the best for image classification tasks, On the other hand, the middle layers are the best for the place recognition and image retrieval tasks. Another evidence was presented in (Sünderhauf et al., 2015), where the AlexNet CNN was used, and it has been found that middle layers can handle the changes in appearance much better than using the higher layers, while the higher layers are the best option to deal with the viewpoint changes.
The work of (Fu et al., 2016; Du and Cai, 2016) has focused on the performance of the last layer of CNN which is the layer that gives the final decision. SVM and Softmax are the two frequently layers that can be used as the last layer, and the result of this study showed that SVM has slightly outperformed the Softmax.

All the aforementioned works have used general pre-trained CNN architectures, i.e., these models were not trained specifically for the place recognition task. The work of (Zhou et al., 2017) can be considered as the first work that attempted to solve this problem by collecting a dataset called Places, which is consists of 10 million images of places in a different environmental situation from around the world. Then, some well-known architectures like AlexNet (Krizhevsky et al., 2012), GoogLeNet (Szegedy et al., 2015) and VGG16 (Simonyan et al., 2014) were trained on this dataset, and as expected, it has outperformed the previous pre-trained models on ImageNet (Krizhevsky et al., 2012) dataset.
Another dataset called Specific PlacEs Dataset (SPED) was collected using nearly 30000 outdoor cameras from all around the world. This dataset contains images from

days and nights through several months and several years attempting to include all possible changes under different conditions (Chen et al., 2017a). This work presented two pre-trained models, AmosNet and HybridNet, AmosNet was trained on the SPED, while the HybridNet was trained on the ImageNet, then fine-tuned on SPED. The results of this study showed that HybridNet has outperformed AmosNet.

Even though deep features are able to outperform the handcrafted features in most cases, it has been found that it is not efficient in handling the viewpoint changes (Xin et al., 2019). To overcome this problem and as shown in (Xin et al., 2019; Chen et al., 2017b; Khaliq et al., 2018, 2019), researchers have tried to add some additional steps before and after the deep features extraction to compensate this limitation. Encoding features to enhance the ability of deep features to be more robust against the appearance changes was one of these steps. On the other hand, some works such as (Chen et al., 2017b), work on determining the image's landmarks to overcome the viewpoint challenge. The approach of (Chen et al., 2017b), tried to detect the most promising landmarks from the output of the convolutional layers of a pre-trained VGG16 deep neural network. Then, the features extracted from the VGG16 layers are fed into BOW (Sivic et al., 2003) to be encoded. Finally, as each image is represented by a vector, the distance between images was calculated by cosine similarity to find the mutual matching of regions in the images.

A slightly different approach was followed in (Khaliq et al., 2018) where the encoding method was VLAD (Jégou et al., 2010) and detecting the landmarks was done through only one layer of a pre-trained AlexNet365 (Zhou et al., 2017) without need to get a mask from another layer like in (Sivic et al., 2003). This approach outperformed the one with BOW.

Another improved approach was introduced in (Khaliq et al., 2019) by the same authors where more features from different layers of the CNN are gathered to improve the performance. In this approach, the HybridNet is the architecture used because of its superiority over other architectures. In addition, the Cosine distance between all images was used to find the best match. This approach worked nearly in real-time, just like the one presented in (Xin et al., 2019).

In this chapter we discussed many methods and many algorithms to deal with the VPR problem, some of them used the traditional ways for extracting features, but we can see clearly that the most recent works based deep features were get remarkable results that was not achieved before.

In this thesis, we have inspired by these works, where deep features are used. Furthermore, the sequence-based methods were proved as a better way for dealing with long-term scenarios and that what was made the DTW a proper algorithm to be used as the core of our localization algorithm. In the following chapter, more details about the proposed approach is given.

CHAPTER 4

DTW BASED ENCODED DEEP FEATURES

In this chapter, the proposed approach is presented. Briefly, the approach start by extracting features through the layers of the deep networks, then, these features are encoded with FV, followed by DTW algorithm, which is used to find the path that gives the best performance through all other possible paths, where the result is the matches between the test images and reference images. It worth mentioning that the improved version of the Fisher Vector (IFV) is used in our approach and it was abbreviated as Fisher Vector (FV) for the rest of this thesis unless another situation was declared.

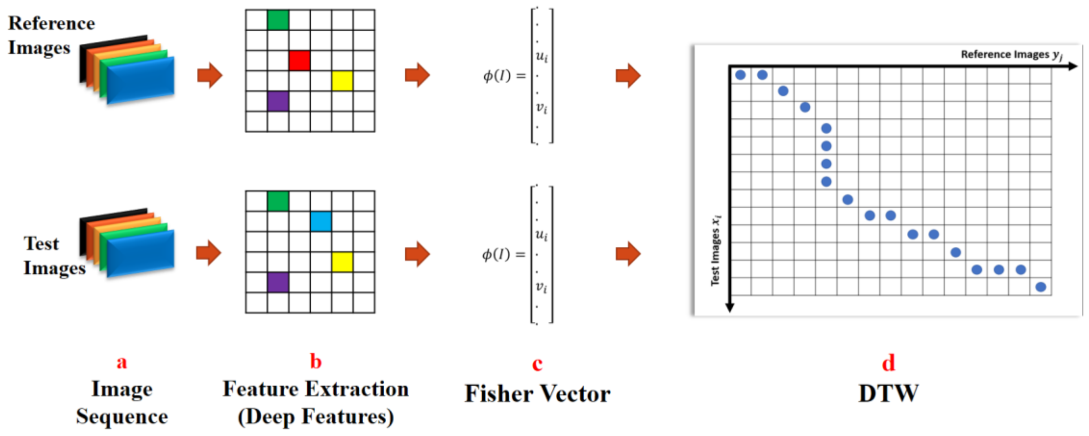


Figure 4.1 The main components of the proposed approach. a) The image sequences (visual map). b) Feature extraction step based deep learning, c) Encoding deep features using Fisher Vector, d) Images alignment and find the best matches using DTW.

4.1 Image Sequence

In this step there are two types of images sequences: Reference Images X and Test Images Y as follows:

$$X = \{x_1, x_2, x_3 \dots, x_n\} \quad \textbf{(4.1)}$$

$$Y = \{y_1, y_2, y_3 \dots, y_m\} \quad \textbf{(4.2)}$$

Where n is the number of reference images and m is the number of test images. The reference and test images should be collected from the same rout. Reference images are fed into the GMM to train a model that can be used later with the test images. This process is explained in detail in section 4.3

4.2 Deep Features Extraction

As explained in CHAPTER 3, there are three common methods that can be used to extract the image's deep features, in this work, the first method was used, i.e., the pixels of each image is fed into the deep network, then, the features are extracted through the output of some layer(s) from the CNN architecture. In more detail, for an image I, the output dimensions of some convolutional layer will be $W \times H \times K$ where W is the width, H is the height and K is the depth of the Feature maps in the selected layer. For such a layer, there are $W \times H$ feature vectors, where each one consists of K components as depicted on Figure 4.2. Let N denote $W \times H$, these features are stacked together in a matrix with size $N \times K$ to be fed into the next stage.

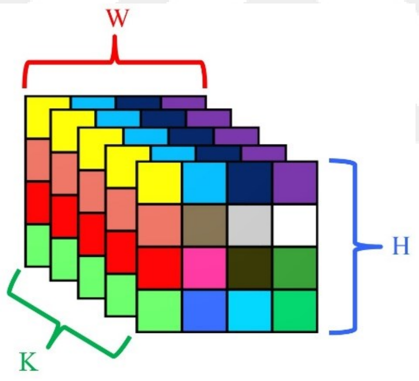


Figure 4.2 An example of a layer in a CNN model. W is the width of the feature map, H is the height of the feature map and K is the number of feature maps in the layer, each group of elements with the same color represents a different feature vector.

In this work, three main CNN architectures were used including VGG16, ResNet50 and HybridNet networks (Zhang et al., 2015; He et al., 2016; Chen et al., 2017a).
Each of these three networks has its own architecture, where the number of convolutional layers and the parameters related to each of these layers are what give a CNN its own specifications and differentiate it from other network models.
VGG16 model consists of 16 convolutional layers, the input should be 224*224*3 i.e., RGB image, this input is passed through the first two convolutional layers with a 64 feature maps for each one and a kernel of size 3*3. The output is reduced with a pooling layer into 112*112*64, this output is fed into the next block which consists of

two layers with a size of 128 feature maps followed by pooling layer reduces the input into 56*56*128. The third block gives an output with size 28*28*256 where each of the three layers in this block consists of 256 feature maps. For the fourth and fifth blocks, there are three layers in each of them with 512 feature maps in each layer, the output of the fourth block is 14*14*512 and the output of the fifth block is 7*7*512. These convolutional layers are followed by three fully connected layers, the first two layers consist of 4096 neurons for each one, and the final layer (output layer) is the decision layer with the SoftMax function.

The ResNet50 model consists of five blocks like VGG16, but with different parameters for each block, the input is a 224*224 RGB image, this image is sub-sampled into 112*112*64 as an output of the first block with a kernel size 7*7, this output is fed into the next block which contains two stages, the first one uses the max pooling with a kernel size of 3*3, then, three convolutional layers with different parameters formulate the second stage where each of them is repeated three times, the first and second layers, each of which has 64 feature maps with a 1*1 kernel size for the first one and a 3*3 for the second. The third one has 256 feature maps and 1*1 kernel size. It is worth mentioning that the kernel size of each layer in this block is applicable for all following blocks because each of them consists of three different types of convolutional layers. The output of the second block is pooled into 56*56*256, this output is fed into the next block which gives an output with size of 28*28*512. In this block, each of the three types of the convolutional layers is repeated 4 times, the first and second ones have 128 feature maps for each and the third one has a 512 feature maps. The fourth block gives an output with 14*14*1024 size, where each convolutional layer is repeated 6 times, the first and second ones contain 256 feature maps and the last one has 1024 feature maps. The last convolutional block gives a 7*7*2048 output with 512 feature maps for the first two convolutional layers and 2048 for the third one with 3 repetitions for each. A decision layer with 1000 neurons and SoftMax function represents the output layer.

The third CNN model is the HybridNet, where it consists of six blocks, each of them has only convolutional layer and pooling layer, the first one has 96 feature maps with a filter size 11*11, the pooling layer gives an output with a size of 55*55*96, this output is reduced in the next block into 27*27*256 where the 256 is the number of the feature maps and a filter with size 5*5 is applied. The third and fourth layers consist of 384 feature maps with a 3*3 filter and a 13*13*384 output size for each of them, the fifth and sixth block have the same number of feature maps (265) with the same kernel size (3*3), but the output of the fifth block is 13*13*256 and for the sixth block is 6*6*256. Two fully connected layers with 4069 neurons for each exist at the end of the HybridNet architecture with a SoftMax function for the second one.

4.3 FV encoding stage

Fisher vector can be produced through two main steps: Building the Gaussian Mixture Model (GMM) and then, obtaining the Fisher Vector (FV).

4.3.1 Training Phase: Building the Gaussian Mixture Model (GMM)

A training dataset (Day left for the Garden point dataset and for berlin_A100 it was the reference sequence used by (Khaliq et al., 2018)) is used to get the main components of the GMM including weight (w_c), mean (μ_c) and covariance (Σ_c) for each cluster (c). These components can be described as follows where M is the number of clusters which is set to 128 clusters:

$$\lambda = \{w_c , \mu_c, \Sigma_c\} \quad c = 1, \dots, M \tag{4.3}$$

4.3.2 Testing Phase: Obtaining the Fisher Vector (FV)

This phase is to extract the FV for both, the training and testing datasets (Day Right, Night Right for Garden Point, and the same testing sequence used in (Khaliq et al., 2018) for (berlin_A100)) based the trained GMM from the previous phase to be matched in the next step, as a result, the best path among them can be found. Two components for each element of the feature vector x_t which produced by a layer of a CNN model are calculated for each cluster in the GMM as follows:

$$u_{dc} = \frac{1}{T\sqrt{\pi_c}} \sum_{t=1}^{T} \Pr(c|\, x_t, \lambda)\, \frac{x_{dt} - \mu_{dc}}{\sigma_{dc}} \tag{4.4}$$

$$v_{dc} = \frac{1}{T\sqrt{2\pi_c}} \sum_{t=1}^{T} Pr(c|\, x_t, \lambda) \left[\left(\frac{x_{dt} - \mu_{dc}}{\sigma_{dc}} \right)^2 - 1 \right] \tag{4.5}$$

Where $d = \{1, \dots, K\}$ is a component of x data vector with dimension K represents the number of feature maps which varies according to the selected layer from the CNN model. $Pr(c|\, x_t, \lambda)$ is the posteriori for each cluster in the λ model as follows:

$$Pr(c|\, x_t, \lambda) = \frac{w_c\, g(x_t \,|\mu_c, \Sigma_c)}{\sum_{j=1}^{M} w_j\, g(x_t \,|\mu_j, \Sigma_j)} \tag{4.6}$$

$g(x_t \mid \mu_c, \Sigma_c)$ is the Gaussian density function. As a result, for each image (I), the calculated components are concatenated to formulate the final fisher vector as illustrated in (4.7)

$$\phi(I) = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ u_c \\ \cdot \\ \cdot \\ v_c \\ \cdot \\ \cdot \end{bmatrix} \quad \textbf{(4.7)}$$

The length of this vector is calculated with the equation:

$$Dimension\ of\ FV = M \times K \times 2 \quad \textbf{(4.8)}$$

Where M is 128 as we said previously, and K is the number of feature maps according to the selected layer. An improved version of FV is generated with a square root normalization followed by L2 normalization applied on $\phi(I)$.

As a result, encoded deep features for both, training and testing datasets were produced and ready to be aligned using the DTW, i.e., the next step. The testing phase of the proposed approach is explained in Algorithm 4.1 where CNN_Model is a pre-trained CNN model where the features are extracted from. λ represents the parameters of the trained GMM model (weight, mean and covariance). u_{dc} and v_{dc} are the components of the Fisher Vector. $norm1$ and $norm2$ are the equations (2.31) and (2.32) respectively.

Algorithm 4.1 $DTWBasedEncodedFeatures(X, Y, M, K, CNN\_Model, D, \lambda, \alpha)$

```
1   $n \leftarrow |X|$                                          //number of reference images
2   $m \leftarrow |Y|$                                          //number of test images
3   $dimension\_of\_FV \leftarrow M \times K \times 2$
4   $reference\_FV[] \leftarrow new[n \times dimension\_of\_FV]$
5   $test\_FV[] \leftarrow new[m \times dimension\_of\_FV]$
6   $U[] \leftarrow new[1 \times dimension\_of\_FV]$             // All $u$ values
7   $V[] \leftarrow new[1 \times dimension\_of\_FV]$             // All $v$ values
8   $counter \leftarrow 1$
9   for $i = 1$ to $n$
10      $Ax_i \leftarrow CNN\_Model(x_i)$                        //Features for image $x_i$
11      for $c = 1$ to $M$                                     //c: a GMM cluster
                                                               //M: number of GMM clusters
```

```
12            for $d = 1\ to\ K$                                    //d: number of components in $Ax_i$
                                                                    //K: dimension of $Ax_i$
13                $U(1, counter) \leftarrow u_{dc}(Ax_i, \lambda)$  //$\lambda$: GMM model
14                $V(1, counter) \leftarrow v_{dc}(Ax_i, \lambda)$
15                $counter \leftarrow counter + 1$
16            end for
17        end for
18        $counter \leftarrow 1$
19       $reference\_FV(i,:) \leftarrow [U, V]$
20       $reference\_FV(i,:) \leftarrow norm1(reference\_FV(i,:))$   //The square root normalization
21       $reference\_FV(i,:) \leftarrow norm2(reference\_FV(i,:))$   // The L2 normalization
22    end for
23    for $j = 1\ to\ m$
24        $Ay_j \leftarrow CNN\_Model(y_j)$
25        for $c = 1\ to\ M$
26            for $d = 1\ to\ K$
27                $U(1, counter) \leftarrow u_{dc}(Ay_j, \lambda)$
28                $V(1, counter) \leftarrow v_{dc}(Ay_j, \lambda)$
29                $counter \leftarrow counter + 1$
30            end for
31        end for
32        $counter \leftarrow 1$
33        $test\_FV(j,:) \leftarrow [U, V]$
34        $test\_FV(j,:) \leftarrow norm1(test\_FV(j,:))$
35        $test\_FV(j,:) \leftarrow norm2(test\_FV(j,:))$
36    end for
37    $C \leftarrow AccumulatedMatrix(reference\_FV, test\_FV, D)$   //Generate the Accumulated
                                                                    Matrix in DTW
38    $path \leftarrow BestPath(C)$                                 //Find the best path in the
                                                                    Accumulated matrix
```

4.4 Sequence alignment using DTW

This component takes the preprocessed input stream of visual data (test images) and the visual map (reference images) to generate a belief about the current place. In other words, the distance matrix will be filled out, and, as mentioned before in DTW, matrix C represents the cumulative distance with a slight difference that the sum of the distance between current matching two images and the maximum (not minimum) of the cumulative distances of the neighboring images is calculated, this difference where the maximum was taken instead of the minimum is because the distance measure used in our approach is the cosine similarity, which has a range between [-1, 1] where -1 means no similarity at all and 1 means the best match, and this distance can be described as follows:

$$D(i,j) = Distance(x_i, y_j) = \frac{|Ax_i| \,.\, |Ay_j|}{\|Ax_i\| \cdot \|Ay_j\|} \tag{4.9}$$

$$C(i,j) = D(i,j) + \max \begin{cases} D(i-1,j), \\ D(i,j-1), \\ D(i-1,j-1), \end{cases} \tag{4.10}$$

The details about building the accumulated matrix is described in the Algorithm 4.2.

Algorithm 4.2 $AccumulatedMatrix(X, Y, D)$

```
1   $n \leftarrow |X|$
2   $m \leftarrow |Y|$
3   $C[] \leftarrow new[n \times m]$
4   $C(0,0) \leftarrow 0$                                    //Fil the first element with 0
5   for $i = 1\ to\ n$
6       $C(i,1) \leftarrow C(i-1,1) + D(i,1)$                //Fill the first column
7   end for
8   for $j = 1\ to\ m$
9       $C(1,j) \leftarrow C(1,j-1) + D(1,j)$                //Fill the first row
10  end for
11  for $i = 1\ to\ n$
12      for $j = 1\ to\ m$
13          $C(i,j) \leftarrow D(i,j) + \max\{C(i-1,j),$         //Fill the rest of elements
                                   $C(i,j-1),$
                                   $C(i-1,j-1)\}$
14      end for
15  end for
16  return $C$
```

And the best path (Algorithm 4.3) can be shown as the path that gives the maximum value of the following function:

$$Q(P) = \sum_{l=1}^{L} C(i_l, j_l) \tag{4.11}$$

Finally, the system has a decision on whether it is a prior visited place or a new place. As an example, Figure 1.3 shows the output images using different kinds of features corresponding to the same input image.

Algorithm 4.3 $BestPath(C)$

```
$path[] \leftarrow new\ array$
$i = rows(C)$
$j = columns(C)$
while $(i > 1)$ & $(j > 1)$
    if $i == 1$ then
        $j = j - 1$
    esle if $j == 1$ then
        $i = i - 1$
    else
        if $C(i-1, j) == \max\{C(i-1, j), C(i, j-1), C(i-1, j-1)\}$ then
            $i = i - 1$
        else if $C(i, j-1) == \max\{C(i-1, j), C(i, j-1), C(i-1, j-1)\}$ then
            $j = j - 1$
        else
            $i = i - 1;\ j = j - 1$
        end if
    $path.add((i, j))$
    end if
end while
return $path$
```

CHAPTER 5

EXPERIMENTAL EVALUATION AND ANALYSIS

The main aims of the experiments presented in this chapter are: 1) investigate the proposed DTW place recognition method. 2) Evaluate the performance of using multiple handcrafted features like SIFT, HOG, and LDB, vs. using the deep features extracted from multiple CNN networks like VGG16, ResNet50, and HybridNet. 3) Compare between different encoding algorithms based deep features. 4) Investigate the performance of the DTW vs. SVM after encoding the deep features by FV. 4) Evaluate the proposed approach against the SVM classifier.
In more detail, we firstly studied the efficiency of the DTW algorithm with handcrafted features, particularly SIFT, HOG, and LDB. Then, we investigated the performance of features extracted from different layers of VGG16 (Zhang et al., 2015), ResNet50 (He et al., 2016) and HybridNet (Chen et al., 2017a) networks. As a result of this experiment, we detected the best layer that obtains the best performance when integrated with DTW for place recognition. After that, we compared the performance of the DTW when used with 1) handcrafted features and 2) deep features extracted from the best layer found in the previous experiment. In the fourth experiment, the encoded deep features were evaluated based multiple encoding algorithms to find the best one that can be used in the next experiment where the performance of the encoded deep features against the deep features without encoding for both classifiers, DTW and SVM were investigated. In the last experiment, the performance of the proposed DTW algorithm was compared with the SVM classifier as one of the most famous classifiers to be used with CNN architectures.

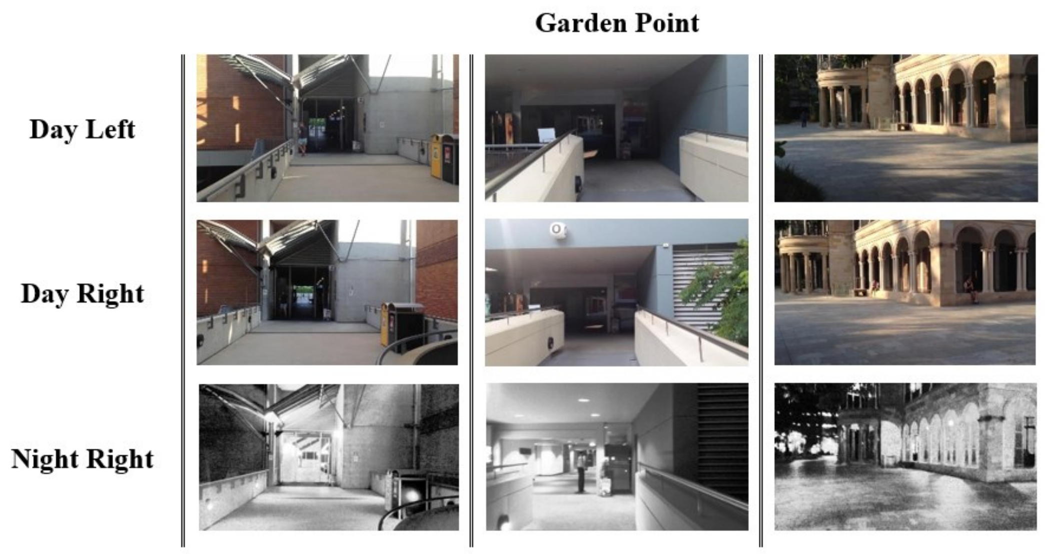
5.1.1 Datasets and Evaluation

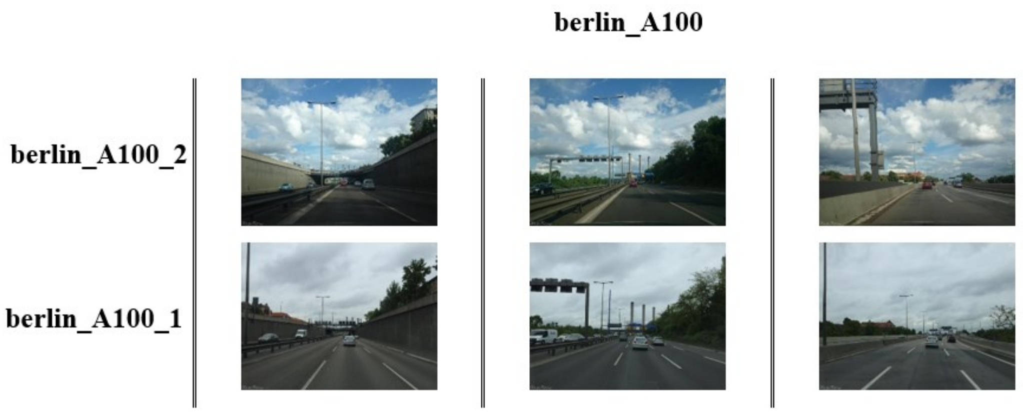
In this study both well-known datasets “Garden Point” and “berlin_A100” (Sünderhauf et al., 2015) were used, and the details are summarized below.

1) The “Garden Point”: which is a dataset that captures the changes in the

pose and lightening conditions in the "QUT" campus. It consists of three sub-datasets: Day left, Day right and Night right. The first two sequences were collected during the day, but with different viewpoints. In addition, the third one has a very close viewpoint to the second one, but it differs in the illumination, and the images of this series where taken at night. Each of these series has 200 images labeled by referring to the corresponding images.

2) The "berlin_A00": which is a dataset collected from a platform called Mapillary where images of the same route were collected by different users with a variation in viewpoint and appearance. In this work, we have used the sub-dataset of "berlin_A00" that was constructed by (Khaliq et al., 2018) where the reference set is consisted of 85 images and the test set consisted of 81 images. In addition, the ground truth of this sub-dataset was made by matching the images which have the same position in terms of GPS.





Error! Reference source not found.. Shows some image samples from the G arden point and the berlin_A100 dataset.

Related to the performance evaluation, the precision-recall curve (PRC), Area under curve (AUC), and the average precision (AP) measures were used.

1) The precision-recall curves can be obtained after finding the best match frame for each test frame among all the frames in the training sequence. The precision (P) is calculated as $P = \frac{TP}{(TP + FP)}$ while the recall (R) is calculated as $R = \frac{TP}{(TP + FN)}$.

   Note that a match is considered as a positive if $D(i, j) < t$, where t is a predefined threshold, otherwise it is considered as a negative match.

   It is worth mentioning here that the threshold (t) is the number of frames between the matched image from reference images and the ground truth.

2) AUC which is the area under PRC: AUC can be calculated using the trapezoidal rule

$$AUC = \sum_{i=1}^{n-1} \frac{p_i^{min} + p_{i+1}^{max}}{2} (r_{i+1} - r_i) \tag{5.1}$$

   Where p is the precision value and r is the recall value, where for each recall value there could be many precision value, so, p_i^{min} is the minimum precision corresponding to r_i and p_i^{max} is the maximum precision corresponding to r_i and n is the considered number of recalls.

3) AP which is the weighted mean of precision was used for each threshold in the PRC, and it can be calculated as

$$AP = \sum_n (R_n - R_{n-1}) P_n \tag{5.2}$$

   Where R_n and P_n are the recall and precision respectively obtained for the threshold n in the PRC.

5.1.2 Dynamic Time Warping with Handcrafted features

Firstly, the performance of the proposed visual place recognition method has been tested after integrating some well-known handcrafted descriptors, in particular, HOG, SIFT, and LDB. The experiment has been initially conducted by matching the selected two sequences (Day left and Day right) from Garden Point. In the first part of this experiment, the cosine similarity matrix was directly initiated, then the matched image with the minimum distance is selected (without DTW). In the second part of this experiment, the matching selection was achieved using the DTW algorithm. The PR curve resulted from both parts is depicted in Figure 5.1. As a result, using DTW has outperformed and improved the performance of all used handcrafted descriptors. It is worth mentioning that a threshold of 1 frame has been used where the result is considered as a true positive either if the test image has met with the exact correspondence reference or it is one frame far.

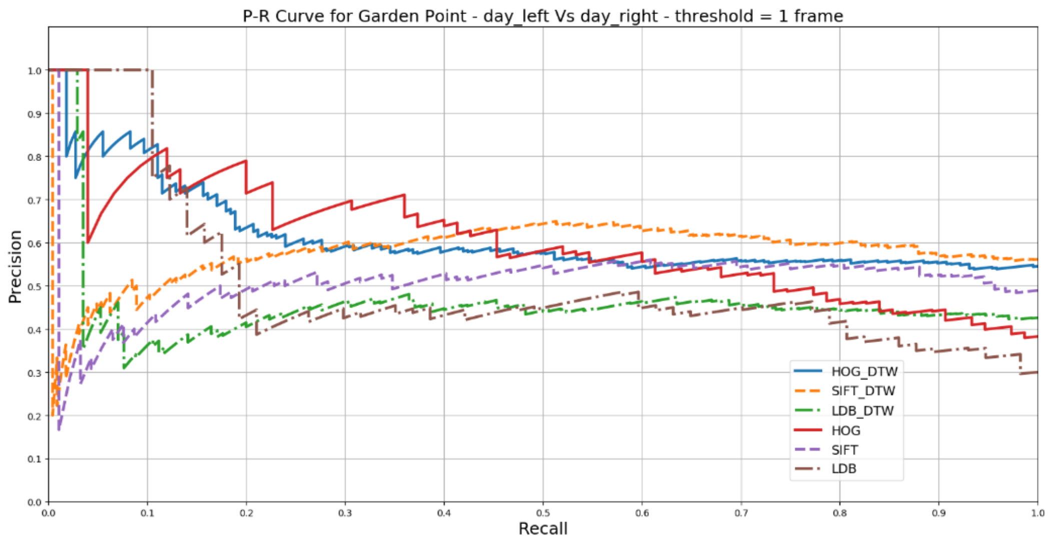


Figure 5.1 Comparing the traditional implementation of the HoG, SIFT and LDB against fed the same features into DTW, using Garden Point (Day left vs Day right) dataset with a threshold equal to 1 frame.

5.1.3 Investigating the performance of VGG16, ResNet50 and HybridNet Layers

In this section, we explore the VGG16, ResNet50 and HybridNet network architectures looking for the layer that achieves the best performance according to the PRC. This experiment has been formulated to find out the layer among the mentioned architectures that achieves the best performance when integrated with DTW. In more detail, the output of each layer was injected into DTW to get the best path between the test and reference images. The last layers from block 3, 4 and 5 were selected to represent the VGG16 and ResNet50. For HybridNet the Convolutional layers from 1 to 6 were selected. In addition, the “Garden Point” dataset has been used in this experiment, where the “day left” and “day right” are the reference and test series respectively, and the threshold was set to 1 frame. The resulted PRC are shown in Figure 5.2, Figure 5.3, and Figure 5.4 for VGG16, ResNet50, and HybridNet respectively. According to this experiment, it could be said that the layer from the last block 5 has outperformed other layers for both VGG16 and ResNet50. Related to the

HybridNet, like other architectures, we can see that "Conv 6", i.e., the last convolutional layer has achieved the best performance among other layers.

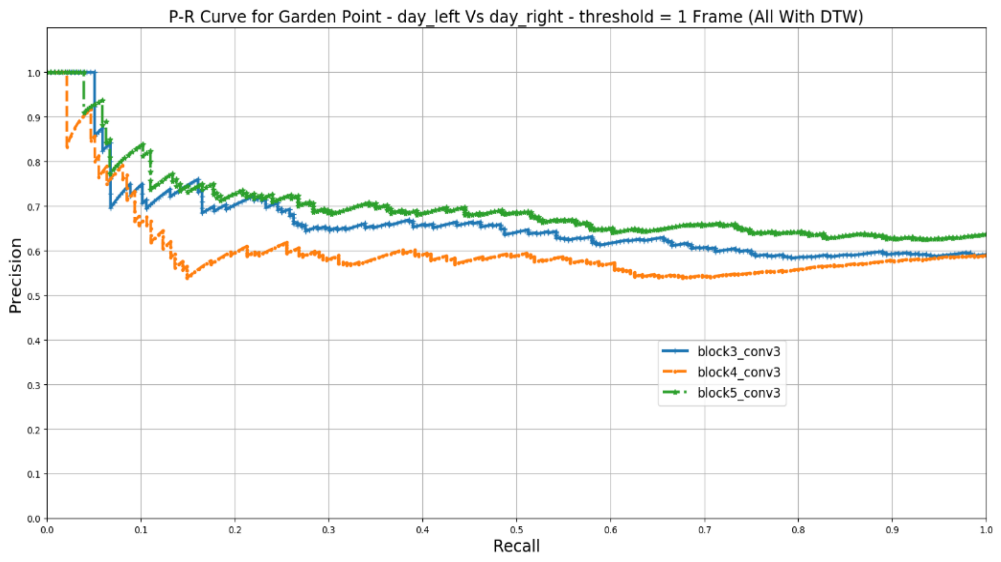


Figure 5.2 PRC for three different convolutional layers of VGG16.

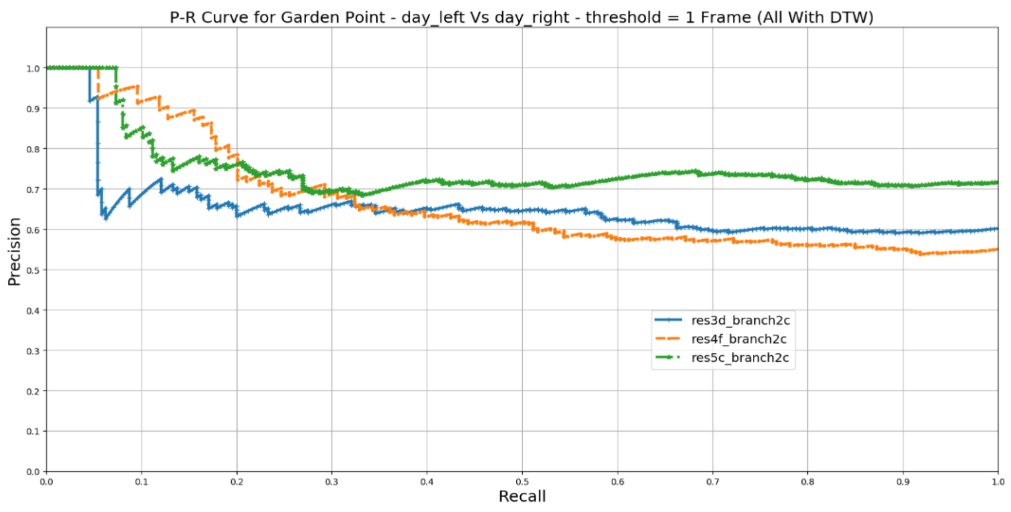


Figure 5.3 PRC for three different convolutional layers of ResNet50.

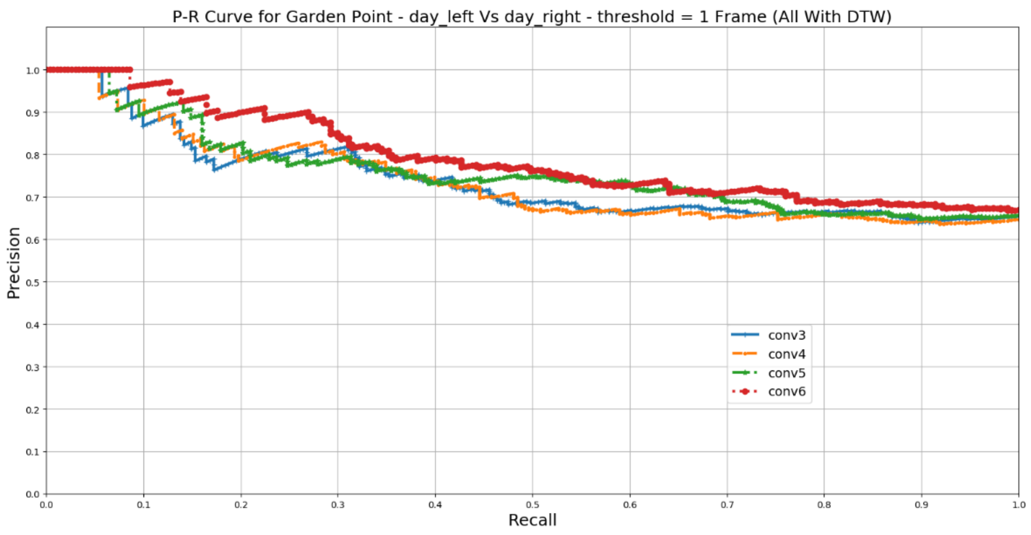


Figure 5.4 PRC for four different convolutional layers of HybridNet.

5.1.4 The Performance of Deep Features

In this experiment, the deep features extracted from the ResNet50's best layer, i.e., the layer that has obtained the highest performance in the previous section, have been evaluated against the used handcrafted features, i.e., HOG, SIFT, and LDB. As shown in Figure 5.6, the deep features have outperformed all other features and was able to obtain high precision over all recall values.

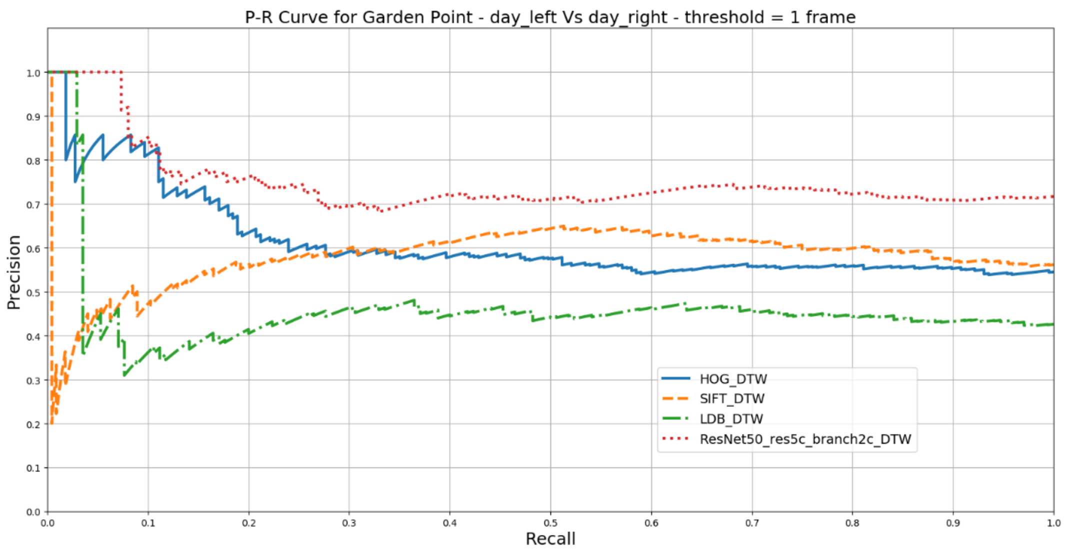


Figure 5.5 PRC for the HoG, SIFT and LDB handcrafted features vs. the layer (res5c_branch2c) from ResNet50, all integrated with DTW. Garden Point (Day left vs Day right) dataset used with a threshold 1 frame.

5.1.5 Performance of different Encoding Schemes

In this experiment, a comparison between BOW, VLAD, Normal FV and Improved FV was made to pick up the most suitable one to be integrated with DTW. Garden Point (Day left vs Day right) dataset was used and deep features were extracted through the third convolutional layer of the block four "block4_conv3" in VGG16 network. This experiment shows that the Improved FV has outperformed other encoding techniques and it is the best to be used in the next experiments as depicted on figure Figure 5.6. In the following experiments, the improved Fisher Vector is abbreviated as Fisher Vector (FV).

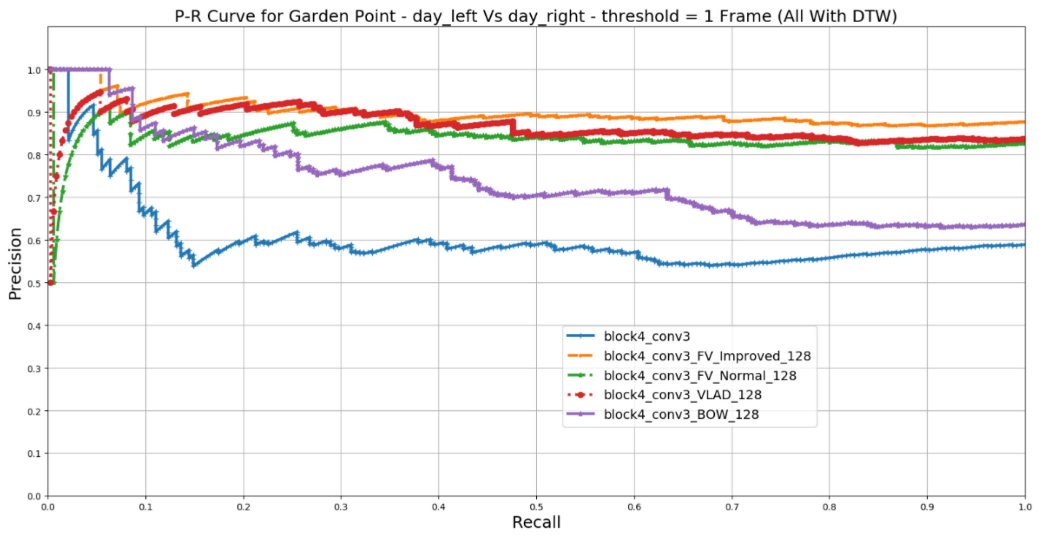


Figure 5.6 PRC for BOW, VLAD, Normal FV and Improved FV integrated with DTW. "block4_conv3" layer from VGG16 network used. All encoding algorithms has a visual codebook with a 128 cluster. Garden Point (Day left vs Day right) dataset used with a threshold 1 frame.

5.1.6 Performance of DTW based Fisher Vector

In this section, the Garden Point and berlin_A100 datasets were used to evaluate the DTW based deep features encoded with the fisher vector against the non-encoded features. In addition, the same scenario was repeated using the SVM classifier based deep features. In addition, the Pre-trained VGG16 and ResNet50 networks were used. Furthermore, the number of GMM was set to 128 for all experiments done in this section.

Overall, the following can be observed:

1) Using the Garden Point (Day left vs Day right), as shown in (Figure 5.7, Figure 5.8, Figure 5.9, Figure 5.11, Figure 5.12, Table 5.1, Table 5.5, Table 5.6), Using the FV leads to improve the performance of a) all the used convolutional layers and b) the used architecture. But for the last two fully connected layers in VGG16, applying the encoding step leads to a huge decrease in the performance which make this kind of layers not suitable to be encoded with fisher vector.
2) Using the Garden Point (Day left vs Night right), as shown in (Figure 5.9, Figure 5.13, Table 5.3, Table 5.7), whenever the DTW was used as the classifier, the features extracted from VGG16 and encoded with FV outperforms the same deep features without FV. However, when the SVM is used as the classifier, the layer from the block 3 encoded through FV was outperformed by the same block without the FV.
3) Using the berlin_A100 as shown in (Figure 5.10, Figure 5.14, Table 5.4, Table 5.8) when the DTW was used as a classifier, the features with FV outperformed the features without, but for the SVM, only features from the block 3 with FV outperformed the same layer without FV.

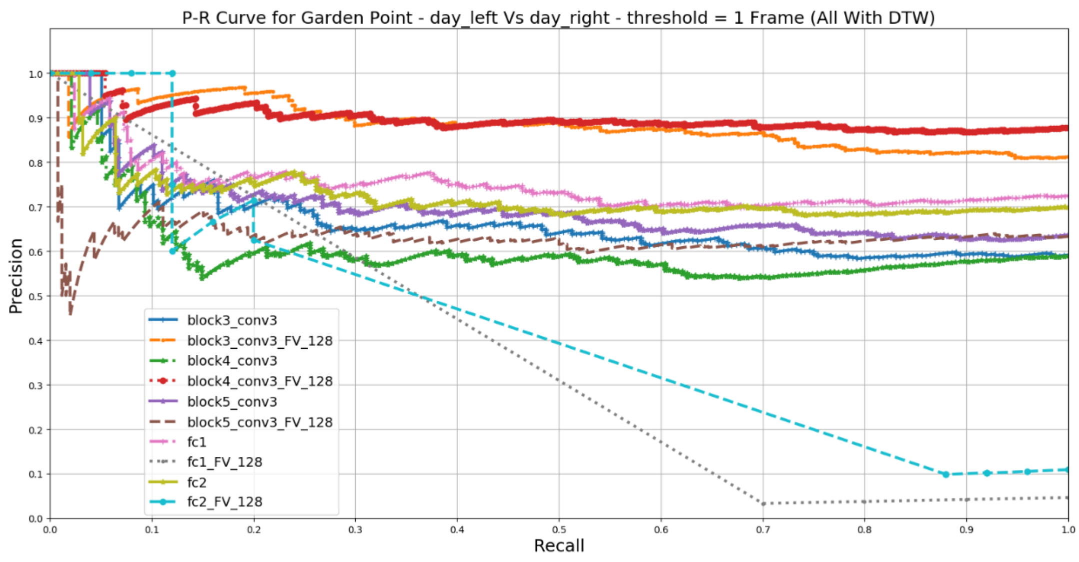


Figure 5.7 PRC for the convolutional layers of VGG16 without FV against the same layers encoded with FV and all are integrated with DTW. Garden Point (Day left vs Day right) dataset used with a threshold 1 frame.

Table 5.1 AUC and AP for layers in VGG16 without FV against the same layers encoded with FV and all are integrated with DTW. Garden Point (Day left vs Day right) dataset used with a threshold 1 frame.

	AUC		**AP**	
VGG16 Layers	**Without FV**	**With FV**	**Without FV**	**With FV**
block3_Conv3	0.664	0.883	0.663	0.883
block4_Conv3	0.604	0.899	0.603	0.898
block5_Conv3	0.699	0.632	0.698	0.630
fc1	0.751	0.373	0.751	0.036
fc2	0.723	0.431	0.723	0.255

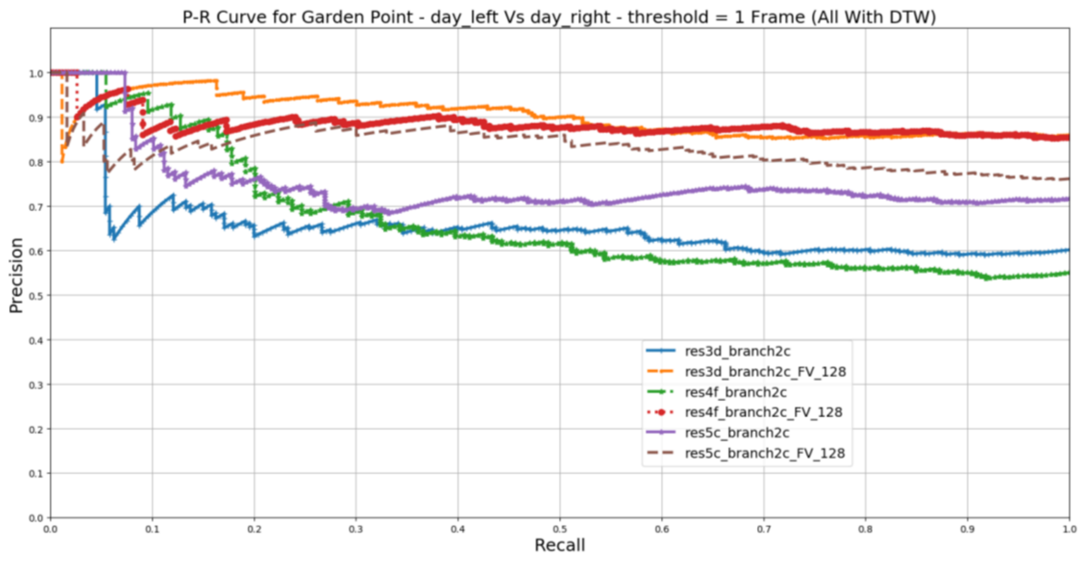


Figure 5.8 PRC for convolutional layers in ResNet50 without FV against the same layers encoded with FV and all are integrated with DTW. Garden Point (Day left vs Day right) dataset used with a threshold 1 frame.

Table 5.2 AUC and AP for convolutional layers in ResNet50 without FV against the same layers encoded with FV and all are integrated with DTW. Garden Point (Day left vs Day right) dataset used with a threshold 1 frame.

	AUC		AP	
ResNet50 Layers	**Without FV**	**With FV**	**Without FV**	**With FV**
res3d_branch2c	0.654	0.902	0.652	0.901
res3f_branch2c	0.667	0.884	0.666	0.883
res5c_branch2c	0.750	0.830	0.749	0.829

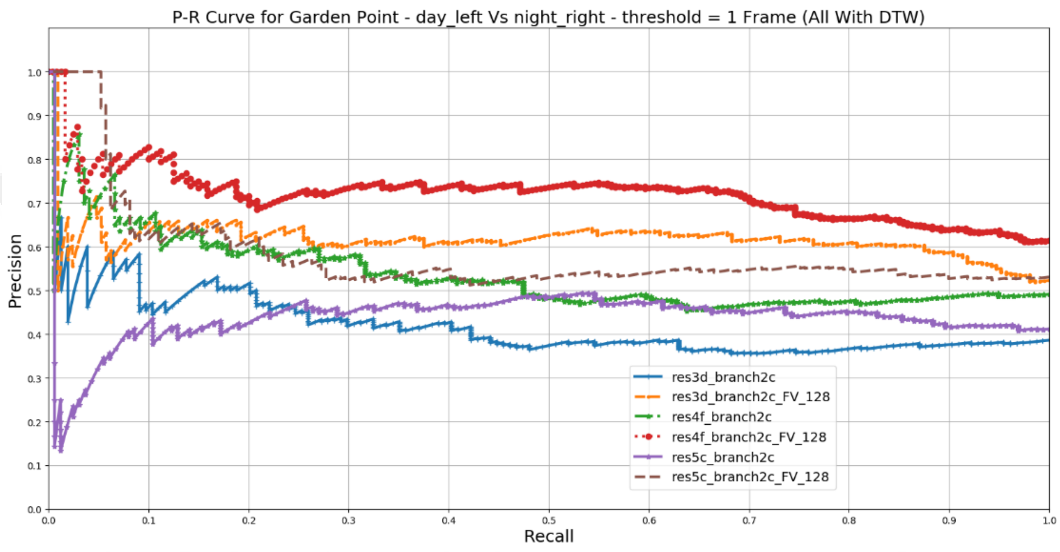


Figure 5.9 PRC for convolutional layers in ResNet50 without FV against the same layers encoded with FV and all are integrated with DTW. Garden Point (Day left vs Night right) dataset used with a threshold 1 frame.

Table 5.3 AUC and AP for convolutional layers in ResNet50 without FV against the same layers encoded with FV and all are integrated with DTW. Garden Point (Day left vs Night right) dataset used with a threshold 1 frame.

	AUC		AP	
ResNet50 Layers	**Without FV**	**With FV**	**Without FV**	**With FV**
res3d_branch2c	0.421	0.619	0.418	0.617
res3f_branch2c	0.538	0.722	0.535	0.721
res5c_branch2c	0.443	0.583	0.439	0.581

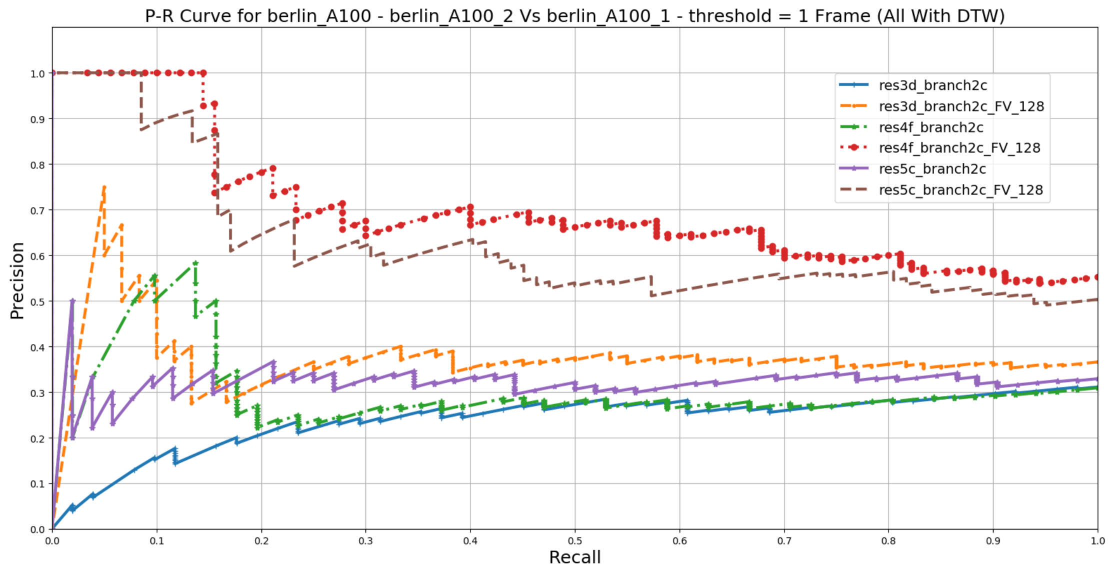


Figure 5.10 PRC for convolutional layers in ResNet50 without FV against the same layers encoded with FV and all are integrated with DTW. berlin_A100 dataset used with a threshold 1 frame.

Table 5.4 AUC and AP for convolutional layers in ResNet50 without FV against the same layers encoded with FV and all are integrated with DTW. berlin_A100 dataset used with a threshold 1 frame.

	AUC		**AP**	
ResNet50 Layers	**Without FV**	**With FV**	**Without FV**	**With FV**
res3d_branch2c	0.247	0.397	0.240	0.372
res3f_branch2c	0.315	0.699	0.299	0.697
res5c_branch2c	0.333	0.627	0.319	0.624

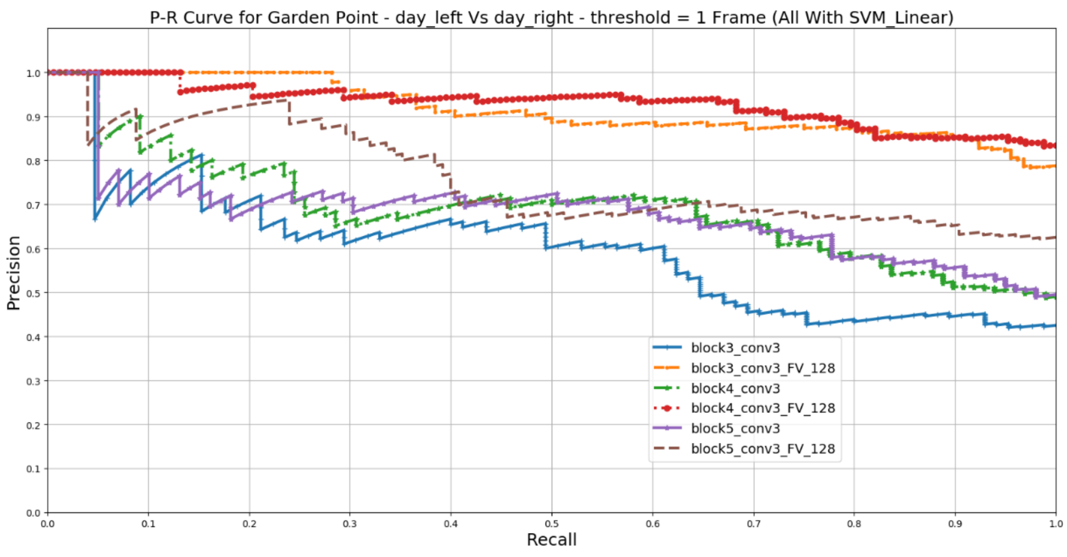


Figure 5.11 PRC for convolutional layers in VGG16 without FV against the same layers encoded with FV and all are integrated with SVM. Garden Point (Day left vs Day right) dataset used with a threshold 1 frame.

Table 5.5 AUC and AP for convolutional layers in VGG16 without FV against the same layers encoded with FV and all are integrated with SVM. Garden Point (Day left vs Day right) dataset used with a threshold 1 frame.

	AUC		**AP**	
VGG16 Layers	**Without FV**	**With FV**	**Without FV**	**With FV**
block3_Conv3	0.601	0.916	0.597	0.916
block4_Conv3	0.693	0.930	0.690	0.929
block5_Conv3	0.683	0.760	0.680	0.759

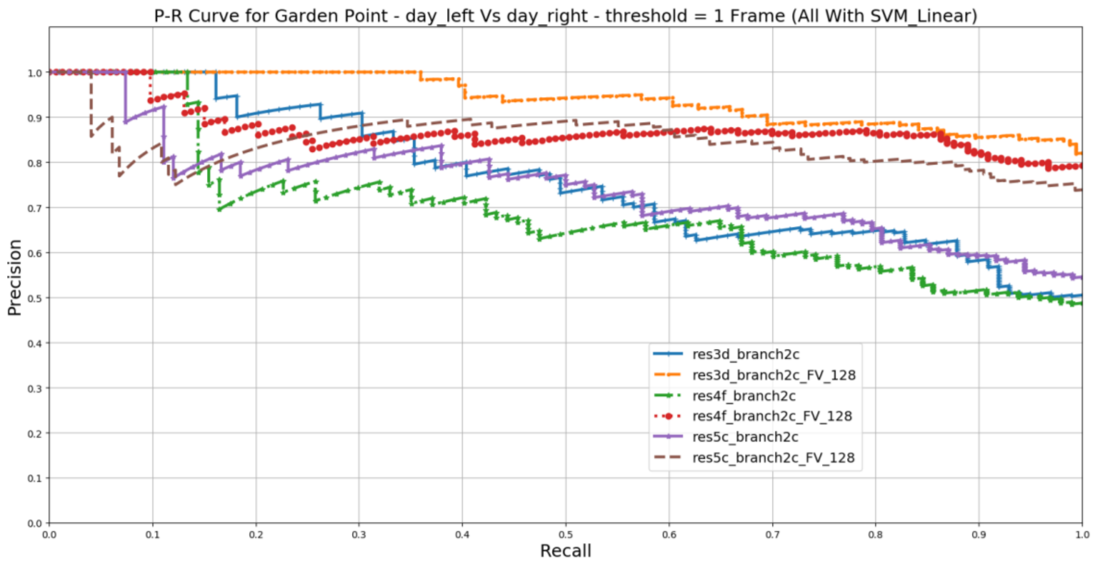


Figure 5.12 PRC for convolutional layers in ResNet50 without FV against the same layers encoded with FV and all are integrated with SVM. Garden Point (Day left vs Day right) dataset used with a threshold 1 frame.

Table 5.6 AUC and AP for convolutional layers in ResNet50 without FV against the same layers encoded with FV and all are integrated with SVM. Garden Point (Day left vs Day right) dataset used with a threshold 1 frame.

	AUC		**AP**	
ResNet50 Layers	**Without FV**	**With FV**	**Without FV**	**With FV**
res3d_branch2c	0.765	0.941	0.764	0.941
res3f_branch2c	0.693	0.872	0.691	0.872
res5c_branch2c	0.747	0.844	0.745	0.843

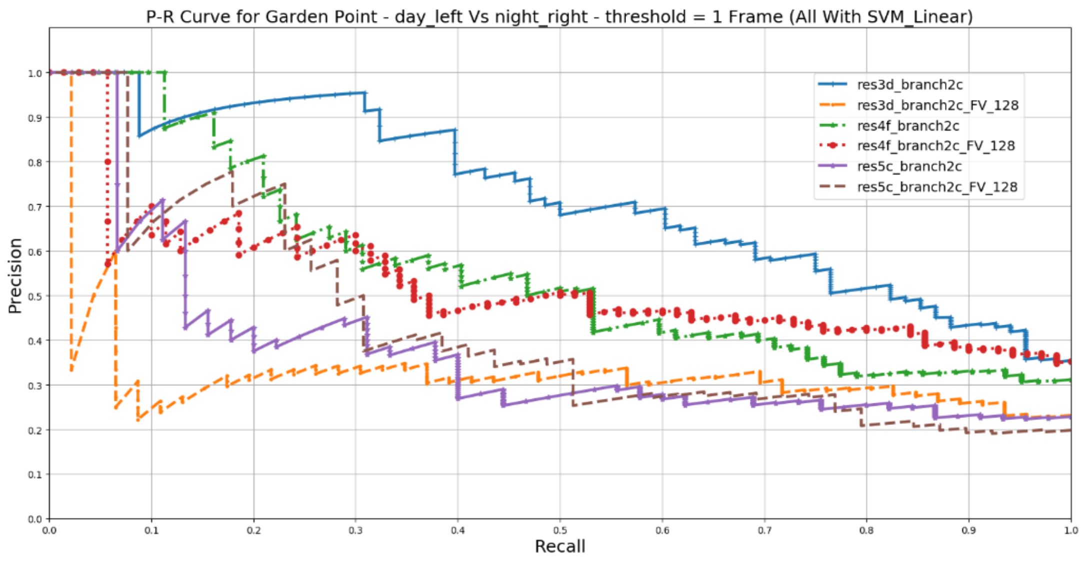


Figure 5.13 PRC for convolutional layers in ResNet50 without FV against the same layers encoded with FV and all are integrated with SVM. Garden Point (Day left vs Night right) dataset used with a threshold 1 frame.

Table 5.7 AUC and AP for convolutional layers in ResNet50 without FV against the same layers encoded with FV and all are integrated with SVM. Garden Point (Day left vs Night right) dataset used with a threshold 1 frame.

	AUC		**AP**	
ResNet50 Layers	**Without FV**	**With FV**	**Without FV**	**With FV**
res3d_branch2c	0.724	0.331	0.721	0.322
res3f_branch2c	0.556	0.530	0.552	0.526
res5c_branch2c	0.380	0.431	0.374	0.424

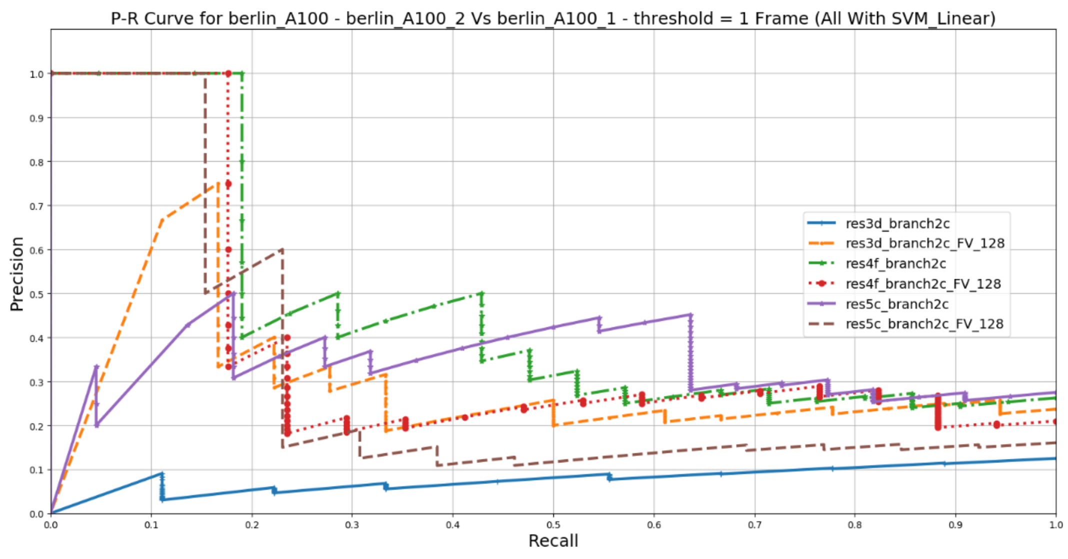


Figure 5.14 PRC for convolutional layers in ResNet50 without FV against the same layers encoded with FV and all are integrated with SVM. berlin_A100 dataset used with a threshold 1 frame.

Table 5.8 AUC and AP for convolutional layers in ResNet50 without FV against the same layers encoded with FV and all are integrated with SVM. berlin_A100 dataset used with a threshold 1 frame.

	AUC		**AP**	
ResNet50 Layers	**Without FV**	**With FV**	**Without FV**	**With FV**
res3d_branch2c	0.090	0.332	0.078	0.282
res3f_branch2c	0.464	0.389	0.455	0.380
res5c_branch2c	0.361	0.317	0.333	0.306

5.1.7 DTW against other Approaches

In this experiment, the performance of the DTW for place recognition is compared with the SVM based algorithm. The results are shown in Table 5.9, and it can be summarized as follows.

a) Using the Garden Point (Day left vs Day right), where the challenge is only the viewpoint, the SVM was able to outperform the DTW using two of the three used layers, i.e., "res3d_branch2c" and "res5c_branch2c".
b) Using the Garden Point (Day left vs Night right) and berlin_A100 datasets, which have viewpoint, appearance and illumination challenges, the DTW can significantly outperform the SVM.

Table 5.9 AUC results for convolutional layers in ResNet50 with FV based DTW against SVM using the Garden Point (Day left vs Day right), Garden Point (Day left vs Night right) and berlin_A100.

	AUC		
CNN	**Conv layer**	**DTW**	**SVM**
Garden Point Day left Day right	res3d_branch2c	0.901	**0.941**
	res3f_branch2c	**0.883**	0.872
	res5c_branch2c	0.829	**0.843**
Garden Point Day left Night right	res3d_branch2c	**0.619**	0.331
	res3f_branch2c	**0.722**	0.530
	res5c_branch2c	**0.583**	0.431
berlin_A100	res3d_branch2c	**0.397**	0.332
	res3f_branch2c	**0.699**	0.389
	res5c_branch2c	**0.627**	0.317

CHAPTER 6

CONCLUSION

The new visual place recognition method presented in this work, integrated the dynamic time warping (DTW) algorithm, to match the current frame from a test sequence to a priori annotated reference sequence frame. This algorithm has been adapted with the features extracted from a deep convolutional neural network (CNN), encoded by the Improved Fisher Vector (IFV). The matching is achieved by the construction of a cost function that measures the distances between the frames in both sequences. Then, an optimal path is found using DTW.

In our experiments, the handcrafted features with DTW outperformed the same handcrafted features without DTW. For the deep features, multiple layers of the VGG16, ResNet50 and HybridNet models were investigated to find the layer that performs better with the DTW algorithm. We found that almost all studied layers gave comparable results, however, the last conventional layer has advantages over other layers when processing some images that have variation between the two sequences in the viewpoint such as Garden Point (Day left vs Day right).

When comparing the deep features against the handcrafted features, even though deep features require more power and memory consumption, it was able to give higher precision for all recall values. In addition, a higher precision was obtained by encoding the deep features with the (IFV) especially for the middle layers of the CNN models where more general features are existing in these layers.

The performance of the DTW and SVM when the FV encoding scheme is used also investigated. The experimental results show superior performance for our approach especially with the challenging datasets in terms of viewpoint and appearance, however, for the viewpoint problem, using the Garden Point (Day left vs Day Right), SVM was able to get a little bit better performance. On the other hand, SVM was not robust enough to face the challenges existed in other datasets like Garden Point (Day

left vs Night Right) and berlin_A100, and for such dataset, there is a clear advantage of our approach as shown in the related experiments.
Further improvements and as future work, some additional steps such as improving the feature extraction procedure, where instead of taking the output of the feature maps directly, a Region of Interest (ROI) can be detected and the feature vectors can be extracted accordingly. This step should make the features more robust against the viewpoint changes. Also, improving the proposed approach to work in real-time is a very important step, this can be done through two steps: creating a general GMM model that can be used for extracting the FV, which can handle any test dataset or live images without a need for re-training. Hence, this can be done by training the GMM on a wide range of datasets collected under different conditions. The second step is to create a new version of the DTW that has the ability to align the incoming images with the reference images.

CHAPTER 7

REFERENCES

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., & Süsstrunk, S. (2012). SLIC superpixels compared to state-of-the-art superpixel methods. IEEE transactions on pattern analysis and machine intelligence, **34(11)**, 2274-2282.

Arroyo, R., Alcantarilla, P. F., Bergasa, L. M., Yebes, J. J., & Gámez, S. (2014, June). Bidirectional loop closure detection on panoramas for visual navigation. In 2014 IEEE Intelligent Vehicles Symposium Proceedings (pp. 1378-1383). IEEE.

Bay, H., Tuytelaars, T., & Van Gool, L. (2006, May). Surf: Speeded up robust features. In European conference on computer vision (pp. 404-417). Springer, Berlin, Heidelberg.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., ... & Zhang, X. (2016). End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316.

Chandrasekhar, V., Lin, J., Liao, Q., Morere, O., Veillard, A., Duan, L., & Poggio, T. (2017, April). Compression of deep neural networks for image instance retrieval. In 2017 Data Compression Conference (DCC) (pp. 300-309). IEEE.

Chen, Z., Jacobson, A., Sünderhauf, N., Upcroft, B., Liu, L., Shen, C., ... & Milford, M. (2017, May). Deep learning features at scale for visual place recognition. In 2017 IEEE International Conference on Robotics and Automation (ICRA) (pp. 3223-3230). IEEE.

Chen, Z., Lam, O., Jacobson, A., & Milford, M. (2014). Convolutional neural network-based place recognition. arXiv preprint arXiv:1411.1509.

Chen, Z., Liu, L., Sa, I., Ge, Z., & Chli, M. (2018). Learning context flexible attention model for long-term visual place recognition. IEEE Robotics and Automation Letters, **3(4)**, 4015-4022.

Chen, Z., Maffra, F., Sa, I., & Chli, M. (2017, September). Only look once, mining distinctive landmarks from convnet for visual place recognition. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 9-16). IEEE.

Cummins, M., & Newman, P. (2008). FAB-MAP: Probabilistic localization and mapping in the space of appearance. The International Journal of Robotics Research, **27(6)**, 647-665

Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection.

Du, K., & Cai, K. Y. (2016). Comparison research on iot oriented image classification algorithms. In ITM Web of Conferences (Vol. 7, p. 02006). EDP Science

Fu, R., Li, B., Gao, Y., & Wang, P. (2016, October). Content-based image retrieval based on CNN and SVM. In 2016 2nd IEEE International Conference on Computer and Communications (ICCC) (pp. 638-642). IEEE.

Glover, A. J., Maddern, W. P., Milford, M. J., & Wyeth, G. F. (2010, May). FAB-MAP+ RatSLAM: Appearance-based SLAM for multiple times of day. In 2010 IEEE international conference on robotics and automation (pp. 3507-3512). IEEE.

Hafez, A. A., Tello, A., & Alqaraleh, S. (2019, April). Visual Place Recognition by DTW-based sequence alignment. In 2019 27th Signal Processing and Communications Applications Conference (SIU) (pp. 1-4). IEEE.

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

Hong, S., Kim, J., Pyo, J., & Yu, S. C. (2016). A robust loop-closure method for visual SLAM in unstructured seafloor environments. Autonomous Robots, **40(6)**, 1095-1109.

Jégou, H., Douze, M., Schmid, C., & Pérez, P. (2010, June). Aggregating local descriptors into a compact image representation. In CVPR 2010-23rd IEEE Conference on Computer Vision & Pattern Recognition (pp. 3304-3311). IEEE Computer Society.

Kate, R. J. (2016). Using dynamic time warping distances as features for improved time series classification. Data Mining and Knowledge Discovery, **30(2)**, 283-312.

Khaliq, A., Ehsan, S., Milford, M., & McDonald-Maier, K. (2018). A holistic visual place recognition approach using lightweight CNNs for severe viewpoint and appearance changes. arXiv preprint arXiv:1811.03032.

Khaliq, A., Ehsan, S., Milford, M., & McDonald-Maier, K. (2019). CAMAL: Context-Aware Multi-scale Attention framework for Lightweight Visual Place Recognition. arXiv preprint arXiv:1909.08153.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

Lowe, D. G. (1999, September). Object recognition from local scale-invariant features. In iccv (Vol. 99, No. 2, pp. 1150-1157).

Lowry, S., Sünderhauf, N., Newman, P., Leonard, J. J., Cox, D., Corke, P., & Milford, M. J. (2015). Visual place recognition: A survey. IEEE Transactions on Robotics, **32(1)**, 1-19.

Milford, M. J., & Wyeth, G. F. (2012, May). SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights. In 2012 IEEE International Conference on Robotics and Automation (pp. 1643-1649). IEEE.

Milford, M. J., Schill, F., Corke, P., Mahony, R., & Wyeth, G. (2011, May). Aerial SLAM with a single camera using visual expectation. In 2011 IEEE International Conference on Robotics and Automation (pp. 2506-2512). IEEE.

Mur-Artal, R., Montiel, J. M. M., & Tardos, J. D. (2015). ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE transactions on robotics, **31(5)**, 1147-1163.

Naseer, T., Spinello, L., Burgard, W., & Stachniss, C. (2014, June). Robust visual robot localization across seasons using network flows. In Twenty-Eighth AAAI Conference on Artificial Intelligence.

Perronnin, F., Sánchez, J., & Mensink, T. (2010, September). Improving the fisher kernel for large-scale image classification. In European conference on computer vision (pp. 143-156). Springer, Berlin, Heidelberg.

Petitjean, F., Forestier, G., Webb, G. I., Nicholson, A. E., Chen, Y., & Keogh, E. (2016). Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm. Knowledge and Information Systems, **47(1)**, 1-26.

Sakoe, H., & Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. IEEE transactions on acoustics, speech, and signal processing, **26(1)**, 43-49.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Sivic, J., & Zisserman, A. (2003, October). Video Google: A text retrieval approach to object matching in videos. In null (p. 1470). IEEE.

Sünderhauf, N., Shirazi, S., Dayoub, F., Upcroft, B., & Milford, M. (2015, September). On the performance of convnet features for place recognition. In 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS) (pp. 4297-4304). IEEE.

Sünderhauf, N., Shirazi, S., Jacobson, A., Dayoub, F., Pepperell, E., Upcroft, B., & Milford, M. (2015). Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free. Proceedings of Robotics: Science and Systems XII.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., ... & Tang, X. (2017). Residual attention network for image classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3156-3164).

Wang, P., Liu, L., Shen, C., Huang, Z., van den Hengel, A., & Tao Shen, H. (2017). Multi-attention network for one shot learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2721-2729).

Xin, Z., Cui, X., Zhang, J., Yang, Y., & Wang, Y. (2019). Real-time visual place recognition based on analyzing distribution of multi-scale cnn landmarks. Journal of Intelligent & Robotic Systems, **94(3-4)**, 777-792.

Yang, X., & Cheng, K. T. T. (2013). Local difference binary for ultrafast and distinctive feature description. IEEE Transactions on Pattern Analysis and Machine Intelligence, **36(1)**, 188-194.

Yue-Hei Ng, J., Yang, F., & Davis, L. S. (2015). Exploiting local features from deep networks for image retrieval. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops (pp. 53-61).

Zhang, X., Zou, J., He, K., & Sun, J. (2015). Accelerating very deep convolutional networks for classification and detection. IEEE transactions on pattern analysis and machine intelligence, **38(10)**, 1943-1955.

Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., & Torralba, A. (2017). Places: A 10 million image database for scene recognition. IEEE transactions on pattern analysis and machine intelligence, **40(6)**, 1452-1464.