A Benchmark Comparison of Visual Place Recognition Techniques for Resource-Constrained Embedded Platforms

Author:
Rose POWER

Supervisor:
Dr. Shoaib EHSAN

A thesis submitted in fulfillment of the requirements for the degree of Masters of Science

in the

Embedded and Intelligent Systems Laboratory
School of Computer Science and Electronic Engineering

UNIVERSITY OF ESSEX

March 15, 2023

Abstract

Autonomous navigation has become a widely researched area of expertise over the past few years, gaining a massive following due to its necessity in creating a fully autonomous robotic system. Autonomous navigation is an exceedingly difficult task to accomplish in and of itself. Successful navigation relies heavily on the ability to self-localise oneself within a given environment. Without this awareness of one's own location, it is impossible to successfully navigate in an autonomous manner. Since its inception Simultaneous Localization and Mapping (SLAM) has become one of the most widely researched areas of autonomous navigation. SLAM focuses on self-localization within a mapped or un-mapped environment, and constructing or updating the map of one's surroundings. Visual Place Recognition (VPR) is an essential part of any SLAM system. VPR relies on visual cues to determine one's location within a mapped environment.

This thesis presents two main topics within the field of VPR. First, this thesis presents a benchmark analysis of several popular embedded platforms when performing VPR. The presented benchmark analyses six different VPR techniques across three different datasets, and investigates accuracy, CPU usage, memory usage, processing time and power consumption. The benchmark demonstrated a clear relationship between platform architecture and the metrics measured, with platforms of the same architecture achieving comparable accuracy and algorithm efficiency. Additionally, the Raspberry Pi platform was noted as a standout in terms of algorithm efficiency and power consumption.

Secondly, this thesis proposes an evaluation framework intended to provide information about a VPR technique's useability within a real-time application. The approach makes use of the incoming frame rate of an image stream and the VPR frame rate, the rate at which the technique can perform VPR, to determine how efficient VPR techniques would be in a real-time environment. This evaluation framework determined that CoHOG would be the most effective algorithm to be deployed in a real-time environment as it had the best ratio between computation time and accuracy.

Acknowledgements

With great appreciation and gratitude to Dr Shoaib Ehsan for giving me the opportunity to perform my Masters studies at the Embedded Intelligent Systems (EIS) Lab at the University of Essex, and for his supervision and guidance over the course of this project.

I would like to thank Dr Klaus McDonald-Maier and Dr Michael Milford, who have provided me with guidance and feedback over the course of my research, accumulating into this thesis.

Additionally, I would like to thank Mubariz Zaffra and Bruno Ferrarini for their continuing support and assistance in my work over the course of the time I have been with the EIS lab.

Contents

List of Figures

List of Tables

List of Abbreviations

VPR	**V**isual **P**lace **R**ecognition
SLAM	**S**imultaneous **L**ocalization **A**nd **M**apping
RMF	**R**eal-Time **M**atched **F**rames
DOF	**D**egree **O**f **F**reedom
FPS	**F**rames **P**er **S**econd
GPS	**G**lobal **P**ositioning **S**ystem
AUC	**A**rea **U**nder the **C**urve
ROI	**R**egions **O**f the **I**nterest
LiDAR	**Li**ght **D**etection **A**nd **R**anging

Chapter 1

Introduction

1.1 Visual Place Recognition

When navigating, it is essential to be able to self-localise. It is this localization that provides the backbone of any and all calculations or estimations. While the solution to this for most modern technology, Global Positioning System (GPS), is appropriate for many modern-day devices, it is unsuitable for use in robots. This is largely due to the fact that GPS is both unreliable and, at times, inaccurate. These problems are often the result of any number of extenuating circumstances, including but not limited to; signal blocking or reflecting from buildings, trees and other objects, radio interference and satellite coverage gaps. Due to the unreliable and inaccurate nature of GPS, it would be beneficial that a different system is put in place to allow robots to not only map but also self-localise.

An alternative method for mapping and localisation is to use Visual Place Recognition (VPR). VPR is most commonly expressed as; when given an image of a location, a human, creature or robot can identify whether it has seen the presented location previously. VPR is generally achieved by first extracting features or landmarks from the image, and then assigning descriptors to either features or the image as a whole. By comparing these features, it can be determined whether the two images are of the same location.

In order for a VPR algorithm to be successful, it must be able to do two things, as demonstrated in figure 1.1. Firstly it must be able to match images taken in the same location but under different visual variations. Secondly it must be able to reject incorrect matches two aliased images of different places.

While it can be argued that VPR shows clear superiority over GPS in its results, it has yet to be determined which method of VPR will produce not only superior results but also the best practical implementation. Lowry et al [1] conducted a detailed survey of the challenges and developments, as well as possible future directions for research on VPR. In addition, Zaffar et al. [2] have proposed an evaluation framework for VPR, as well as a method of quantifying viewpoint and appearance variation.

1.2 Problems facing VPR

While VPR presents the potential for a much more reliable method of self-localisation, it is not without its own shortcomings. One of the greatest shortcomings of VPR is due to significant appearance variations between images of the same location. These variations are largely due to the changing conditions of the environment.

It is widely accepted that there are 4 four types of variation that images can be subject to. These variation types are: seasonal variation, viewpoint variation,

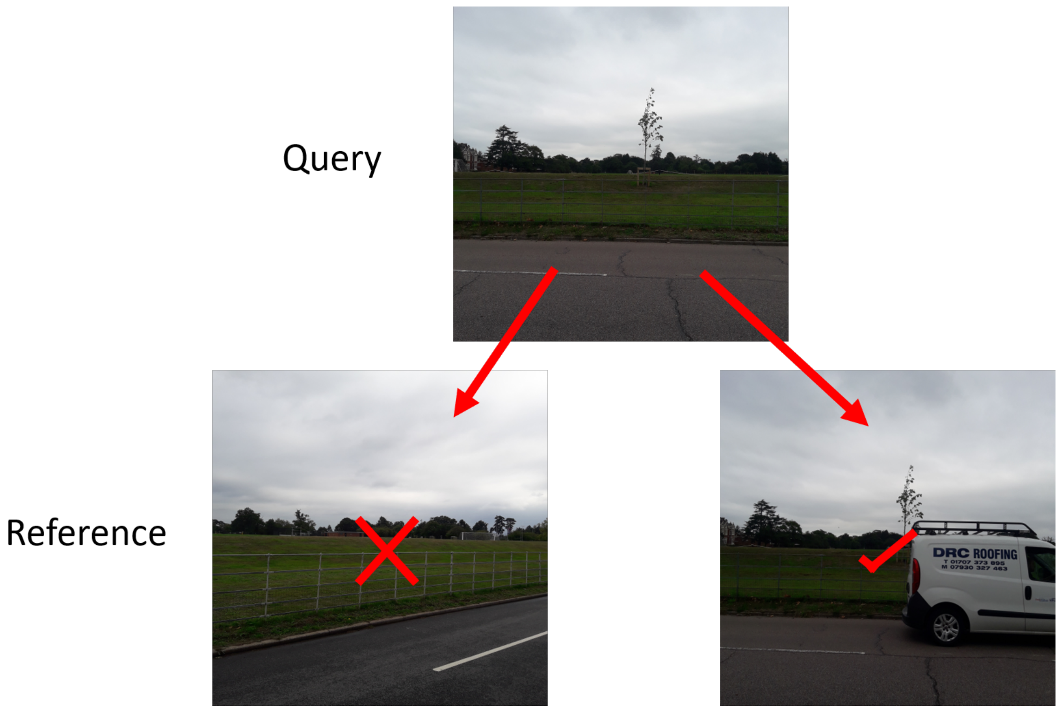

FIGURE 1.1: An example of a query image, as well as a correctly and incorrectly matched reference image from ESSEX3IN1 dataset, representing the difficulties faced by VPR techniques.

dynamic objects and illumination variation, as shown in figure 1.2. Each type of variation poses a unique problem that must be overcome to obtain an accurate and reliable location identification from the VPR system.

Seasonal Variation: The variation that occurs between images taken in different seasons will cause a location to appear significantly different. For example, the height of summer, with lush outdoor greenery, will look very different to the depths of winter, with snowy and bleak scenes [3] [4].

Viewpoint Variation: In many early VPR techniques, it was assumed that images of the same location would be taken at the same angles. However, in reality this is not at all practical, as when moving around, scenes will be viewed from different angles. Viewpoint variation is the image variation when these images capture a scene from different angles [5] [6].

Dynamic Objects: Dynamic objects are objects that move or change, causing a scene to appear different when viewed at different times [7] [8]. Examples of this are cars or other vehicles that, while to us as humans would not inhibit our ability to identify the scene, will cause a street to look significantly different to a VPR algorithm.

Illumination Variation: This occurs when images are taken at different times of the day or under changing artificial illumination conditions [9] [10]. Images taken using a digital camera in low illumination generally experience more noise than under a high illumination [11], unless compensated for by image exposure time. It could be concluded that it is generally more difficult to extract features from images taken in poorly illuminated conditions.

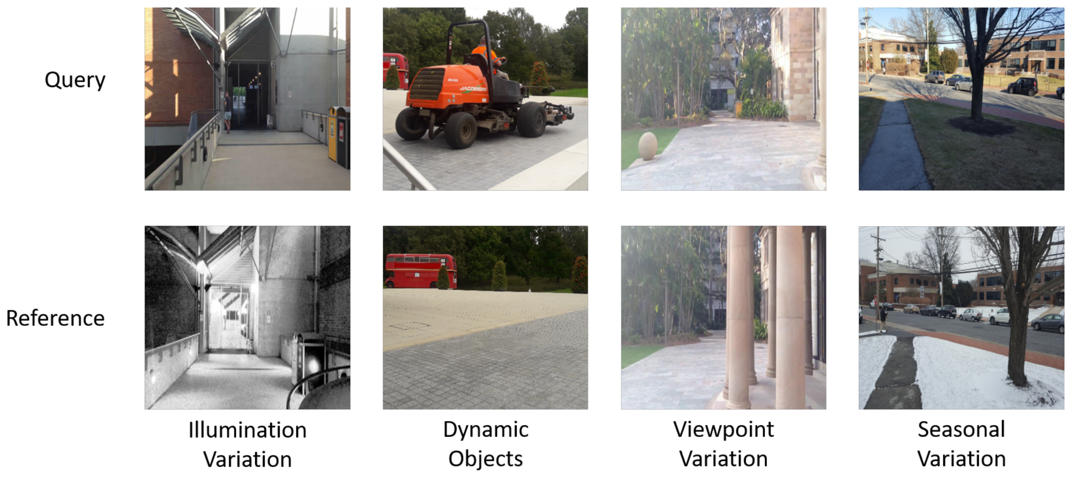

FIGURE 1.2: Examples of query and corresponding reference images from the Campus Loop, ESSEX3IN1 and GardensPointWalking datasets, representing the four different types of variation present in most VPR datasets. These variation types are illumination, season and viewpoint variation, as well as dynamic objects.

1.3 VPR for UAVs

One of the most exciting and rapidly expanding areas of computer science, and robotics in particular, is the use of Unmanned Aerial Vehicles (UAVs). In order to achieve autonomy, UAVs must, like any other robot, be able to track their own location. One of the most promising applications of VPR is its use on UAVs. However, the application of VPR to UAVs is less than simple, as in addition to the aforementioned problems that VPR itself faces, UAVs present their own additional problems for VPR to tackle, that are specific to the platform. Zaffar et al. [12] made a study to determine the effectiveness of using different VPR methods on an aerial platform.

While there is a wide range of sensors that could be carried onboard a UAV, the limited payload of a UAV has to be taken into consideration when choosing which sensors to mount on the UAV. Most powerful modern sensors are too heavy to keep flight energy efficient, meaning UAVs cannot be kept in flight for very long without seriously draining their power. The work described in this thesis focuses on tackling VPR using visual information from a camera, as these are generally small and lightweight enough to be carried onboard a small UAV, and information from internal sensors such as a gyroscope and an accelerometer. This combination of camera and internal sensors is the most commonly used UAV setup for navigation.

One of the greatest problems that UAVs face is extreme viewpoint variation. This viewpoint variation is the same as previously mentioned, only to a greater extent, due to the introduction of a third dimension. In addition, due to the agility of small UAVs, it is very likely for a UAV to approach a scene from very different viewpoints. This is a fatal problem for techniques that use whole-image descriptor representation, and greatly challenging for feature-based techniques. This added dimension, coupled with the 6-degree of freedom (6-DOF) from use of aerial platforms and a wide range of viewpoint approaches causes great problems even for VPR methods that are designed to specifically handle viewpoint variation.

In order to tackle large appearance changes, current feature-based techniques make use of high-quality feature descriptors such as SURF [13] and SIFT [14]. However, these descriptors are generally too computationally expensive to be employed

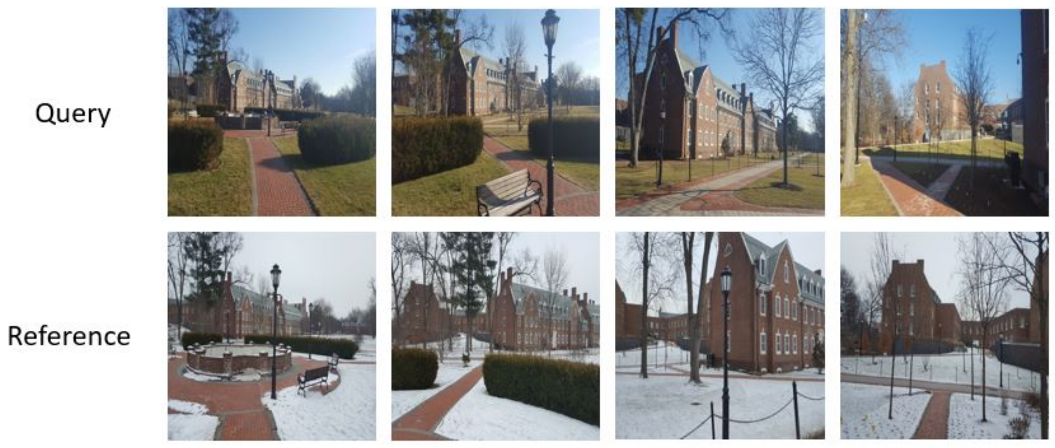

FIGURE 1.3: Examples of query and corresponding reference images from the Campus Loop dataset, which is used for benchmarking throughout the experiments detailed in this report.

on a small UAV. This highlights another problem for the deployment of VPR on small UAVs, their limited computational capabilities. This limited computational capability is problematic due to both the intense computation that is required by most state-of-the-art VPR techniques and the size of the feature descriptors themselves. As such there is a computation overhead limit for many types of VPR systems when attempting to run on small UAVs. This limit can prevent the VPR system from working at full capacity, greatly reducing the resulting recall of the system, or simply preventing the system from working at all. In addition to the computational limitations of feature-based techniques; employing deep-learning techniques usually requires a powerful Graphics Processing Unit (GPU), which typically cannot be carried onboard a small UAV. Maffra et al. [15] [16] [17] make use of binary features that are low cost, making them more computationally suitable for employment on small UAVs. [18] was the first work to make use of binary features for VPR. The results showed, however, that the method was very sensitive to noise.

Maffra et al. [15] introduces and implements a pipeline for performing VPR onboard a small UAV with limited computation power. The proposed pipeline in [15] includes a loop-closure detection algorithm, that does not require prior knowledge of the environment, as well as a method of generating orthophotos from features provided by the SLAM system. Orthophotos are generally constructed using vanishing points, which are estimated by extracting line segments from an image. However, the estimation of vanishing points is difficult due to the small intersection angles between segment lines. In addition, determining which lines are associated to each vanishing point causes errors [19]. [20] uses orthophotos generated by detecting vanishing points to correct for camera rotation when taking images of a location. One of the benefits of the proposed method in [15] is that its method of creating orthophotos does not rely on the estimation of vanishing points but instead estimates the major plane of the scene. However, high precision is only achieved in [15] when orthophotos can be successfully be generated, which often proves difficult as the major plane is often difficult to identify. [21] improves upon the method proposed in [15] to achieve a robust relative position estimation, using drift correction and relocalisation.

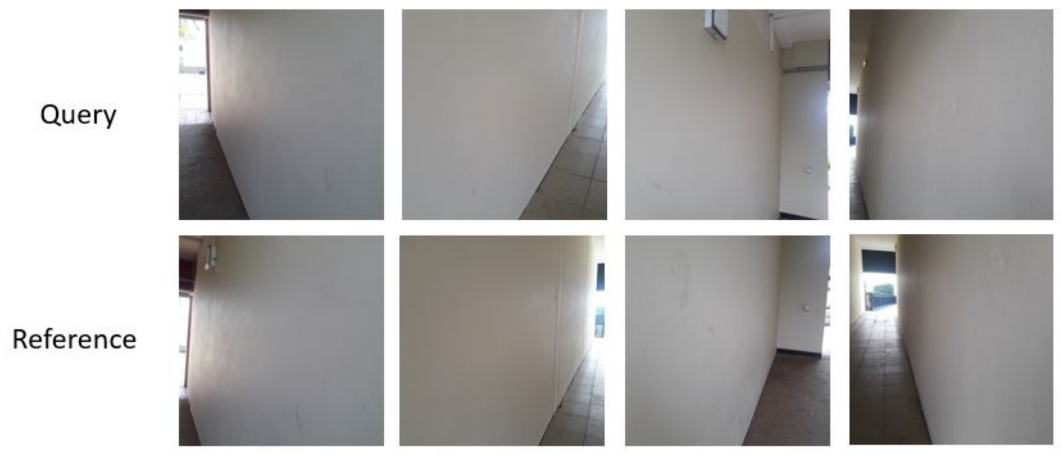

FIGURE 1.4: Examples of query and corresponding reference images from the ESSEX3IN1 dataset, which is used for benchmarking throughout the experiments detailed in this report.

The use of orthophotos was removed in [16], due to the aforementioned problems in their generation. [16] works to improve upon [15], by combining 3D and 2D information, as well as focusing the proposed pipeline on the implemented geometric checks. The BoW [22] approach, which is used in [15], disregards all geometric information. Generally, when employing navigation for ground robots, it is expected that scenes will be viewed up-right, as such the method proposed in [23] is enough to achieve navigation. However, when implementing navigation for UAVs, where it is likely that a scene will be viewed from a wide range of viewpoints, it is essential that geometric checks are implemented to confirm geometrical consistency between images. [16] implements the methods proposed in [24] and [25] for its geometrical checks.

This is further improved in [17] which introduces depth completion by implementing a map densification step to further improve the benefits of combining 3D and 2D information. Allowing for feature-based matching between images with wide viewpoint variation. Many state-of-the-art depth completion algorithms, such as [26] and [27], employ the use of CNNs. However, these approaches require powerful GPUs which, as mentioned before, cannot be carried by small UAVs. On the other hand, CPU-only approaches, such as [28], often rely on densely populated and good quality depth information. Whereas the method proposed in [17] implements a CPU-based approach designed for input form a SLAM system, allowing the handling of sparsely populated maps and a certain amount of noise.

1.4 Research Methodology

This section will discuss the methodology of the work and experiments conducted over the course of my masters.

1.4.1 Research Aims

Overall, the aims of the research conducted over the course of my masters were to look into the use and practical applications of VPR and to provide new insight that would be of value to the computer vision community. To do this we chose to

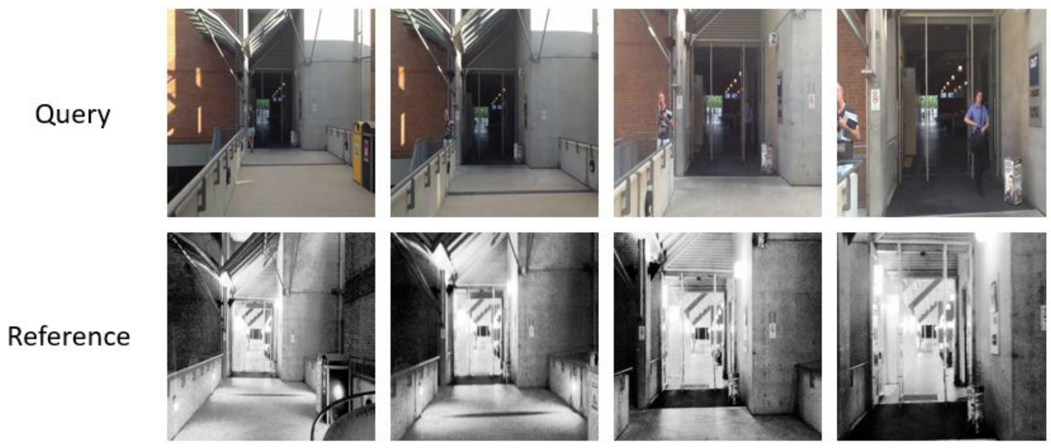

FIGURE 1.5: Examples of query and corresponding reference images from the GardensPointWalking dataset, which is used for benchmarking throughout the experiments detailed in this report.

investigate the effects of hardware on several stare-of-the-art VPR techniques. In doing so we would answer the question, does hardware have an effect on VPR and if so, what?

To investigate this the selected VPR techniques were implemented on several popular embedded platforms, as well as two reference high-end platforms. Originally, we planned to investigate the precision-recall and power consumption of each technique, however this was changed. Instead, we investigate CPU usage, memory usage, processing time, accuracy, average power consumption and our own benchmark, as presented in this thesis, Real-Time Matched Frames (RMF).

1.4.2 Research Motivations

This research was motivated by the work done by Zaffar et al. in [29] and [12]. These works investigate the accuracy, processing power consumption and projected memory requirements of several state-of-the-art VPR techniques. Additional this research was motivated by the work done by Hulens et al. in [30], who investigate the processing speed, power consumption, motor efficiency and flight time of several popular embedded platforms with a view towards on-board UAV image processing.

While accuracy, processing power and memory requirements of VPR techniques have been looked into previously, in the case of Zaffar et al., and the power consumption and processing speed have been looked into, in the case of Hulens et al., they have not been looked into with a view towards a platforms effect on VPR. Therefore, this work is motivated by the desire to fill in the gaps and determine the effects of the architecture and build of several popular embedded platforms on several state-of-the-art VPR techniques.

1.4.3 Research Contributions

The contributions of this thesis can be broken down into two main sections which are as follows:

1. There are many factors that can affect the results of VPR. Hardware factors, often determined by architecture and platform, such as processor speeds and

memory capacity, are constrained when employing autonomous navigation on UAVs. Despite the usefulness in understanding the effects of constrained platforms on VPR, this is a relatively unexplored area. In order to fill this gap in literature, this thesis contributes a hardware-based benchmark evaluation of several state-of-the-art VPR techniques, focusing on the effects of platform architecture on the performance of VPR. This contribution provides new insight into the practicality of using embedded platforms for VPR in robotic systems.

2. To be useful in autonomous navigation, a VPR technique must be able to handle image processing and recognition in real-time. While there are many evaluation metrics that are widely used to evaluate Visual Place Recognition (VPR) techniques, their relevance to real-world scenarios can be ambiguous. Higher results may reflect that a technique retrieves a high percentage of correct matches, but real-world factors like image-retrieval time and platform-speed are often not considered. This means a computationally-intensive technique, that is impractical for use in a real-world or real-time application, may be ranked above other more practical techniques. This thesis contributes a novel benchmark allowing for the evaluation of algorithms with a view to real-world, real-time VPR. This is important because it provides new insight into the use and practicality of VPR in a real-time environment, allowing for better evaluation of VPR algorithms with a specific view towards real-world deployment.

1.4.4 List of Publications

The following contributions were made over the course of this masters:

1. R. Power, M. Zaffar, B. Ferrarini, M. J. Milford, K. D. McDonald-Maier and S. Ehsan, "A Benchmark Comparison of Visual Place Recognition Techniques for Resource-Constrained Embedded Platforms," IEEE Access, 2023.

Chapter 2

Literature Review

This section contains a detailed literature review of papers that I have read over the course of my degree.

2.1 Overview

Visual Place Recognition as a field has developed greatly primarily thanks to its wide spread application in autonomous systems. VPR is essential in any autonomous system that requires the use of localisation. This includes the likes of construction, agriculture, industry and a host of other domains.

While the review presented in this chapter details literature on both SLAM and VPR, it should be noted that these principals are not the same, nor intrinsically necessary for the others use. VPR can be considered a sub-section of SLAM as a method of loop-closure or can be used independently as a its own localisation system [2]. SLAM can make use of other methodologies to achieve loop-closures [31].

The rest of this chapter is as follows; section 2.2 gives a brief overview of the core research within the field of SLAM. Section 2.3 gives an overview of the field of VPR, before section 2.4 details the various techniques that exist for VPR. Sections 2.5 and 2.6 will present the current methods of benchmarking, both VPR and platforms respectively. Section 2.8 will summarise the literature reviewed in this thesis and its basis for the work in this thesis.

2.2 SLAM

Cadena et al. [31] presented a detailed account of SLAM systems. This account details both the software and hardware research pertaining to SLAM. Zaffra et al. [32] details the links between software and hardware in the application of SLAM and how these factors might affect long-term autonomy of a SLAM system. Efficient mapping topologies [33], feature extraction and matching [34], location estimation [35] and loop closure techniques [36] have all been core areas of research within the field of SLAM.

A major driving factor behind VPR research has always been the sensor technologies they rely on. SLAM systems have long taken advantage of the low-cost compact design of acoustic sensors for measuring range. [37] presents an early implementation of this type of system, with other unique implementations presented in [38] [39] [40].

Another sensor technology that has driven SLAM research is Light Detection And Ranging (LiDAR). LiDAR utilises light to measure distance and provides depth images of an environment. This was utilised for SLAM in [41] [42] [43]. The authors of [44] achieve a real-time loop-closure utilising a LiDAR-based system.

One of the most widely utilised sensor technologies for SLAM are cameras. Monocular cameras are favoured due to their low-cost, wide availability, ease of use and information quality. Monocular camera-based SLAM implementations are presented in [45] [46] [47]. The authors of [48] present a comparison of monocular and Stereo SLAM. Stereo camera-based SLAM implementations are presented in [49] [50] [51]. RGB-D sensor-based SLAM implementations are presented in [52] [53], and evaluations of RGB-D based SLAM are presented in [54] [55] [56]. Omni-directional cameras have been used for SLAM due to their wide field of view. Omni-directional camera-based SLAM implementations are presented in [57] [58], and a review of Omni-directional camera-based SLAM is presented in [59]. Event cameras are favoured due to their high dynamic range and are excellent for use in dynamic environments. However, in a static environment they provide minimal visual information. The authors of [60] combine event and monocular cameras for a singular SLAM system.

2.3 VPR

Visual-SLAM refers to a SLAM system where the source of information is a camera. The objective of this system is both to create a map of a previously unknown environment, and to localise a robot within this mapped area. [61] presents a thorough report on Visual-SLAM. Localisation in this system can further be broken down into Visual localisation and Visual Place Recognition for Loop-closure. Visual localisation, also referred to as Visual Odometry, overlaps information from consecutive frames to estimate the movement of a robot. [62] presents a thorough report on Visual Odometry. Visual Place Recognition refers to identifying previously visited locations under appearance and viewpoint variation. Lowry et al. [1] conducted a thorough survey of VPR, investigating the theory behind VPR, its relation to other domains, its challenges and possible future areas of research within the field. VPR has its majority of applications in loop-closure for Visual-SLAM, however, it also has many other applications in the wider field of computer vision, as explored by [2].

2.4 VPR Algorithms

There are two methods for implementing VPR that have been researched. These are feature based and learning based approaches. Feature based methods generally work by first extracting local feature descriptors such as SURF [13] or SIFT [14]. A study was made comparing the accuracy and computational load trade off of local image descriptors in [63]. Once the features are extracted, they are stored for later comparison to query images. One approach that is often employed in state-of-the-art VPR methods for sorting and storing local image descriptors is the Bag of Words (BoW) [22] technique. BoW works by organising descriptors into clusters, depending on the descriptor space each descriptor is closest too. These clusters are then used to form a histogram representation of the image. Another approach often used is VLAD [64]. VLAD is seen as an improvement over BoW in many scenarios. This is because in addition to sorting and storing descriptors, it also records the relative location of each descriptor within their corresponding descriptor space.

Learning based approaches use Convolutional Neural Networks (CNNs) that are pre-trained for image classification to extract layer activations as features. One approach that can be used by learning based VPR methods in Maximum Activations of Convolution (MAC) [65], which uses global max pooling to create its image representation. For this technique, the cosine similarity of two MAC descriptors is used to determine whether two images are the same.

Within the two main method types there are many state-of-the-art VPR methods already developed. [29] makes a study of evaluating a number of state-of-the-art VPR techniques, comparing their accuracy and computational complexity. In addition, the feature encoding and matching times are compared to determine which methods can run in real-time whilst maintaining good precision-recall.

2.4.1 Feature Based

In the early days of VPR techniques, handcrafted high-quality feature descriptors such as SURF [13] and SIFT [14] were employed to label images. These descriptors can be classified as either local or global feature descriptors. SIFT and SURF features were used for VPR in [66] and [67] respectively. SIFT [14] features use difference of gaussians to extract key points from an image to create its descriptors. SURF [13] is a modified version of SIFT which makes use of Hessian-based detectors as opposed to traditional SIFT's Harris detectors.

Histogram-of-orientated-gradients (HOG) [68] [69] is a feature-based technique and was used for VPR in [70]. HOG makes use of gradients that are calculated for every pixel in an image and stores them in a histogram. While HOG is one of the most widely used feature descriptors, its performance lags behind other current state-of-the-art methods. More recently another feature-based technique, CoHOG [71], was developed. CoHOG makes use of HOG descriptors to represent regions of interest (ROI) and are compared using cosine matching in order to achieve lateral viewpoint tolerance.

Seq-SLAM [72] is a feature-based technique that processes sequences of images to find a best matched route by creating a confusion matrix. While this method can take advantage of the use of sequential information to gain excellent tolerance against seasonal and illumination variation, it struggles greatly when exposed to viewpoint variation. In addition, Seq-SLAM is also shown to struggle [29] when a sequence of images is taken by a camera moving at varying speeds.

2.4.2 CNN Based

The use of Convolutional Neural Networks (CNNs), convolution auto-encoders (CAEs) and deep/shallow neural nets have successfully demonstrated superior results for VPR over traditional handcrafted feature-based techniques. Other applications that have also demonstrated a superiority when employing CNNs, CAEs and deep/shallow neural nets include [65] [73].

The use of CNNs for VPR was studied in [74] where a pre-trained CNN was used to extract features from the layers of an image. These features were later used for image comparison in [75]

Using the knowledge provided in [74], Chen et al trained two CNNs [76] on the specific places dataset (SPED). The two CNNS, HybridNet and AMOSNet, both share the same architecture as CaffeNet [77]. While HybridNet uses weights taken directly from CaffeNet and trained on the ImageNet dataset [78], AMOSNet's weights are randomised.

A CNN is traditionally designed for image classification, as such it does not generate the descriptors required for VPR. In [64] a Vectors of Locally Aggregated Descriptors (VLAD) layer was added to a traditional CNN architecture to allow for

TABLE 2.1: Table showing example datasets created for VPR research. The environment, viewpoint variation type and apperance variation type for each dataset are provided.

Dataset Name	Environment	Viewpoint Variation	Appearance Variation
17 Places	Indoor	Lateral	Illumination
24/7 Query	Outdoor	6-DOF	Illumination
Campus Loop	Mixed	Lateral	Seasonal
Corridor	Indoor	Lateral	None
Cross Seasons	Outdoor	Lateral	Seasonal
ESSEX3IN1	Mixed	Lateral	Dynamic Objects
GardensPointWalking	Mixed	Lateral	Illumination
Living Room	Indoor	Lateral	Illumination
Nordland	Outdoor	None	Seasonal
SPEDTest	Outdoor	None	Illumination and Seasonal
Synthia	Synthetic Outdoor	Lateral	Seasonal

end-to-end training. This allowed the authors of [64] to train the CNN specifically for the use of VPR. Other CNN models that implemented a VLAD layer include AlexNet [77] and VGG-16 [79].

Following the introduction of the implementation of a VLAD layer, the design of the layer itself is something that has been extensively studied. The pooling approaches that are employed on convolutional layers is a particular aspect that has been extensively studied. Examples of these pooling approaches that have been research include Max-Pooling [65], Cross-Pooling [73], Sum-Pooling [80] and Spatial Max-Pooling [81].

The use of regions-based descriptions of images was suggested to increase the matching performance of VPR, as demonstrated in [82] [83] [84]. Additionally, in [85] a lightweight CNN-based regional approach and VLAD layer were implemented to form the technique RegionVLAD.

2.5 VPR Benchmarking

Benchmark analysis has a significant impact on the understanding of a model's usability within the target application. One of the most popular VPR benchmarking methods is precision-recall. Recall is the proportion of true positive results that are correctly predicted as positive. This is a desirable benchmark as it reflects the number of true positive results. While this tends not to be highly rated in information retrieval, and is often neglected in Machine learning or Computational linguistics, in the context of Computational Linguistics and Machine Translation, Recall has shown to have great weight in predicting the success of Word Alignment [86]. On the other hand, Precision is the proportion of predicted positives that are true positives. This is what Machine learning, Data mining and Information Retrieval focus on. These two measures, and their combinations, focus on positive examples and predictions. Combined the two measures capture some information about rates and kinds of errors made, however, neither captures any information about how well a model handles negative cases as noted in [87].

ROC was highlighted by Flach [88] for its utility to Machine learning analysis. ROC characterised skew sensitivity for many measures within the field of machine learning, allowing for the utilisation of ROC format to present geometric information of the nature of such measures and their sensitivity to skew. Fürnkranz and Flach

[89] elaborated on this analysis, extending it to an unnormalized PN variant of ROC and targeting their analysis towards rule learning.

Powers et al [90] developed an unbiased accuracy measure to avoid the bias of Precision, Recall and accuracy. Powers defines the concept of Informedness which quantifies how informed a predictor is for a specified condition and specifies the probability that a prediction if informed in relation to a condition. Contrary to this, Markedness quantifies how marked a condition is for the specified predictor, and specifies the probability that a condition is marked by the predictor.

2.6 Platform Benchmarking

In terms of benchmarking resource-constrained embedded platforms the most prominent metrics to take into consideration are power consumption, memory utilization and CPU utilization. An analysis of deep neural networks (DNN) architectures was conducted in [91]. The authors recorded the accuracy, power consumption and memory footprint of each DNN. Additionally, the number of parameters and operations were counted for each computer vision task. The experiments were carried out on an NVIDIA Jetson TX1 board [92].

Huang et al. follow a similar approach in [93] with their exploration of the speed/accuracy trade-off for full image classification. In their experiments, they compare the performance of an Intel Xeon CPU and an NVIDIA Titan X GPU.

TANGO [94] measures inference time, power consumption and memory usage to assess various CNN models implemented on a variety of hardware platforms. These platforms included an embedded GPU and an FPGA. The importance of energy usage was highlighted by Palit et al [95]. They presented an energy estimation model and empirical data from the evaluation of several CNNs.

Fan et al. [96] show that the CPU utilization of running processes is directly related to the power consumption of the CPU. This power consumption becomes a significant factor over extended periods of time within all fields of robotics and especially in battery powered UAVs. The work done in [96] is extended upon by Zaffra et al. [12], who investigate the accuracy, processing power consumption and projected memory requirements of several state-of-the-art VPR techniques, including both feature-based and CNN-based techniques.

2.7 Datasets

A major factor in the growth of the field of VPR is the wide spread availability of open-source datasets. Many of these datasets processes unique challenges, with a variety of viewpoint and appearance variations. The number of images contained within a dataset varies with each. Additionally, the number of images within a dataset can vary from use to use, as often subsets of a dataset are used in experiments. Typically, datasets contain images of the same location but under different viewpoint or appearance variation. Table 2.1 lists some example open-source datasets [97][98][99][100][101][102][103][104][105][82][106] produced for VPR research over the past years. Figure 2.1 shows some examples of images contained within each dataset listed in table 2.1. Additionally, table 2.2 lists the number of images in each of the datasets used later in this thesis.

While there are datasets that contain visual and inertial information publicly available, such as KITTI [107], most traverses mainly exhibit forward camera motion captured from front-facing cameras. This method of capture, however, makes

TABLE 2.2: Table showing the number of images in each of the datasets used in Chapters 3 and 4.

Dataset Name	No. of Images
Campus Loop	100
ESSEX3IN1	210
GardensPointWalking	200

labelling ground-truth very difficult. In [16] new datasets were recorded specifically for the application of place recognition using flying or hand-held setups that have side facing cameras. This allows for clear and accurate labelling of ground-truth. The captured sequences demonstrate both appearance changes and great viewpoint variation. In addition, datasets were recorded that contain extreme viewpoint variation in order to isolate the problem for viewpoint variation.

The 17 Places dataset was introduced in [97] and consists of many different indoor environments, including office, lab, hallway and bedroom environments. The 24/7 Query dataset was introduced in [98], this dataset consists of many sets of three images taken of the same place under different conditions. Each set of images has a corresponding three reference images. Conditions the images were taken under include 6-DOF viewpoint variation, different times of day and dynamic objects. Campus Loop, introduced in [99], is a small scale dataset consisting of traverses of a university campus during different seasons. In addition to seasonal variation, this dataset also contains lateral viewpoint variation between traverses. Corridor was introduced in [100] and is a small scale dataset, consisting of low resolution images. This dataset has some lateral viewpoint variation but no appearance variation. Cross Seasons was introduced in [101], and was built upon the CMU Visual Localisation dataset [108] and the Oxford RobotCar dataset [109]. This dataset contains both seasonal and illumination variation. ESSEX3IN1 was introduced in [102] and consists of a travers of the University of Essex campus. The dataset contains lateral viewpoint variation, as well as dynamic objects and uninformative scenes. Gardens Point Walking was introduced in [103] and is one of the most widely employed datasets for testing VPR. The dataset consists of three traverses, two performed during the day with lateral viewpoint variation, and a third during the night. Living Room was introduced in [104] and consists of high resolution, wide field of view images. The dataset contains viewpoint and illumination variation. The images in this dataset were taken from a home service robot, so have a perspective from close to the floor. The Nordland dataset was introduced in [105] and consists of a train journey through Norway, the journey was performed during four different seasons giving it seasonal variation. The original dataset does not contain any viewpoint variation but some uses of the dataset [99] [29] employed the technique of manually cropping images to produce synthetic lateral viewpoint variation. The SPEDTest dataset was introduced in [82], while it has no viewpoint variation, it has challenging illumination and seasonal variation. The Synthia dataset was introduced in [106] and consists of traverses through different simulated outdoor environments. The traverses contain seasonal variation, as well as some lateral viewpoint variation and dynamic objects.

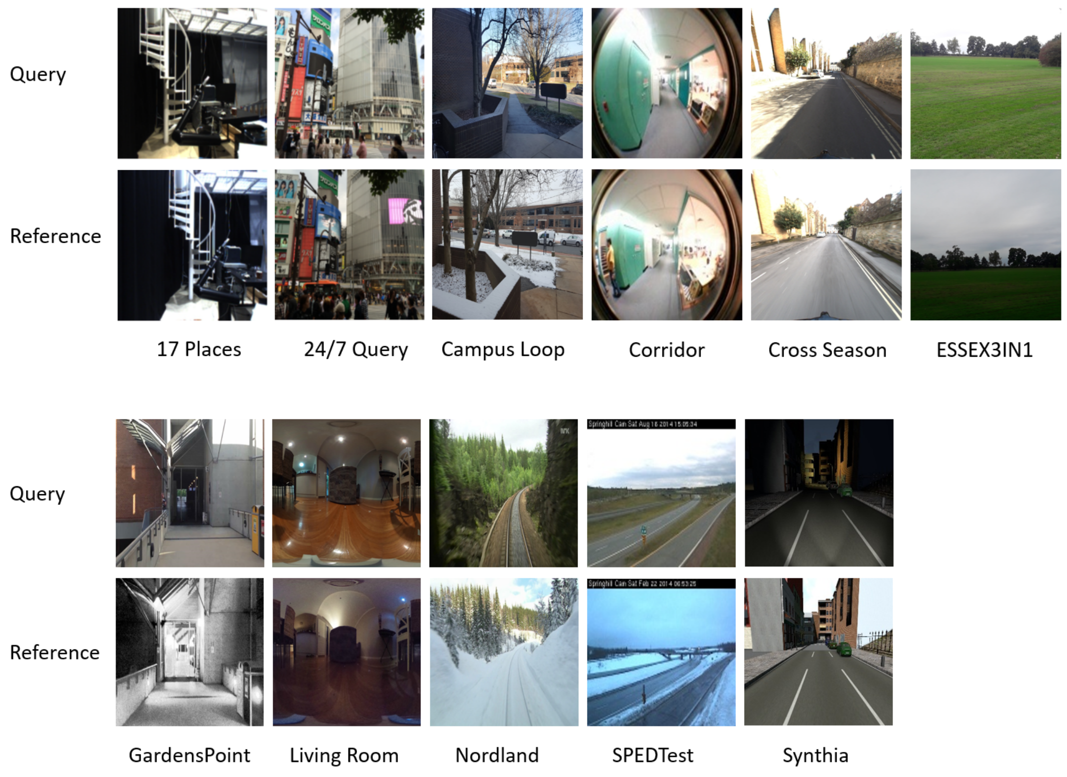

FIGURE 2.1: Figure showing example query and corresponding reference images from various datasets, as listed in table 2.1.

2.8 Summary

This chapter detailed a review of the current research landscape surrounding localisation for autonomous robotic navigation with specific focus on Visual Place Recognition. The points below were covered:

1. An overview of SLAM
2. The relation between Visual-SLAM, Visual Odometry, VPR and other methodologies.
3. A detailed survey of Visual Place Recognition techniques and datasets. This serves as a basis for the work of Chapters 3 and 4.
4. An overview of benchmarking Visual Place Recognition techniques, from both a hardware and software perspective. This serves as a basis for the work in Chapter 3

The majority of existing literature in the field of VPR focuses on proposals of new VPR techniques. These techniques are usually environment specific and computationally expensive. There is limited literature on the comparison and benchmarking of existing VPR techniques. This work is motivated by a desire to fill the limited literature on the subject of benchmarking VPR, as well as the largely unexplored comparison in regards to hardware-based factors such as processor speed and memory capacity. Motivated by the work done by Zaffar et al. [29] [12], and Hulens et al. [30], the work in this thesis attempts to provide a comparison of VPR in regards to hardware limitations with a view to the deployment of VPR onboard lightweight embedded platforms able to be carried by UAVs. The benchmark presented in chapter 3 is based on several key metrics including; place-matching accuracy, image encoding and descriptor matching time, and memory needs.

Additionally, there is little literature investigating the performance of VPR in real-time environments, as would be necessary for real-world applications. There are many evaluation metrics widely used to evaluate Visual Place Recognition (VPR) techniques, however, their relevance to real-world scenarios is limited. In chapter 4 this thesis provides an evaluation benchmark with a view to evaluating VPR algorithms on their usefulness in a real-world environment, and is intended to provide useful information about a VPR technique's usability within a real-time application. This is motivated by a desire to further focus the research done on VPR towards it's real-world deployment.

Chapter 3

VPR Hardware Benchmark

To understand the usability and usefulness of a model, that model must first undergo evaluation on a number of parameters. With the results of this benchmark we can truly understand the usefulness of said model within its target application. This chapter contributes a benchmark analysis of several popular embedded platforms when performing VPR. These techniques are deployed on-board several state-of-the-art embedded platforms in order to evaluate the effects of hardware and architecture on VPR performance, as well as the performance and requirements of the platforms themselves. This is significant because it provides new insight into the practicality of using embedded platforms for VPR in robotic systems and determines what effects platform architecture has on VPR. This benchmark investigates accuracy, CPU usage, memory usage, processing time and power consumption.

3.1 Background

Recalling a previously visited place using only visual information has become a subject of interest within the robotic vision community and therefore Visual Place Recognition (VPR) has developed as a dedicated field within autonomous robotics over the past 15 years [1]. VPR is a fundamental task for autonomous navigation as it enables self-localization within an environment. To be useful in autonomous navigation, a VPR system must be able to run in real-time [17]. There are many factors that can affect VPR. External factors include various appearance and viewpoint variations. Whereas internal factors are usually due to architecture or platform, and as such can include factors such as computational power and memory capacity.

Although robots are often equipped with resource-constrained hardware, the subject of implementing and comparing VPR on resource constrained embedded platforms, which is potentially very useful for the application on VPR onboard small unmanned aerial vehicles, is relatively unexplored. This work presents a hardware-focused benchmark evaluation of a number of state-of-the-art VPR techniques on public datasets. Popular single board computers are considered, including ODroid, UP and Raspberry Pi 3, in addition to a commodity desktop and laptop for reference. Analysis based on several key metrics is presented, including place-matching accuracy, image encoding time, descriptor matching time and memory needs.

3.2 Experimental Setup

3.2.1 Evaluation Datasets

There have been many datasets created for the purpose of testing VPR. This subsection will introduce the datasets used to test the techniques and platforms introduced later in the experimental setup. The number of images in each dataset can be found in Table 2.2.

Garden Point Walking [103]

This dataset was created at the Queensland University of Technology. The first traverse of the dataset was taken during the day and the reference traverse was taken at night from laterally different viewpoints. As such the dataset displays both illumination and viewpoint variation, as well as dynamic objects making it a challenging dataset. Some example images from this dataset are shown in figure 1.5.

Campus Loop [99]

The campus loop dataset was created from a sequence of indoors and outdoors images of a campus environment. The first traverse of the dataset was taken on a snowy day with very cloudy weather and the second traverse was taken several days later when the snow had melted and it was sunny. Whilst the seasonal variation will not cause the images captured indoors to vary greatly, the image captured outside present significant seasonal and some illumination variation. In addition, the images also contain viewpoint variation and dynamic objects. Some example images from this dataset are shown in figure 1.3.

ESSSEX3IN1 [102]

The ESSEX3IN1 dataset was created to provide both viewpoint and appearance variation for the purpose of testing VPR. The dataset contains images that are confusing for VPR techniques, with challenging dynamic objects and uninformative scenes, causing most state-of-the-art techniques to struggle as shown in [102]. The dataset contains 210 images and has frame-to-frame correspondence between query and reference images. Some example images from this dataset are shown in figure 1.4.

3.2.2 VPR Techniques

The experiments in this paper are carried out on a selection of state-of-the-art VPR methods, as a representation of a variety of technique classes.

HOG

Histogram-of-orientated gradients (HOG) [68] [69] is one of the most widely used VPR techniques that uses hand-crafted feature descriptors. This technique calculates a gradient for every pixel in an image and organises these gradients into the bins of a histogram. These bins will contain the sum of the gradient magnitudes. The technique then uses the cosine function to compare the query and reference images. While HOG is one of the most widely used techniques, it does not perform to the same level as any other state-of-the-art techniques. While this performance level means that HOG is not at the forefront of any new leading research, it makes HOG a good technique for use in making comparisons. For the implementation of HOG, we use a cell size of 8*8, a block size of 16*16, and total of 9 histogram bins as suggested by the authors in [29].

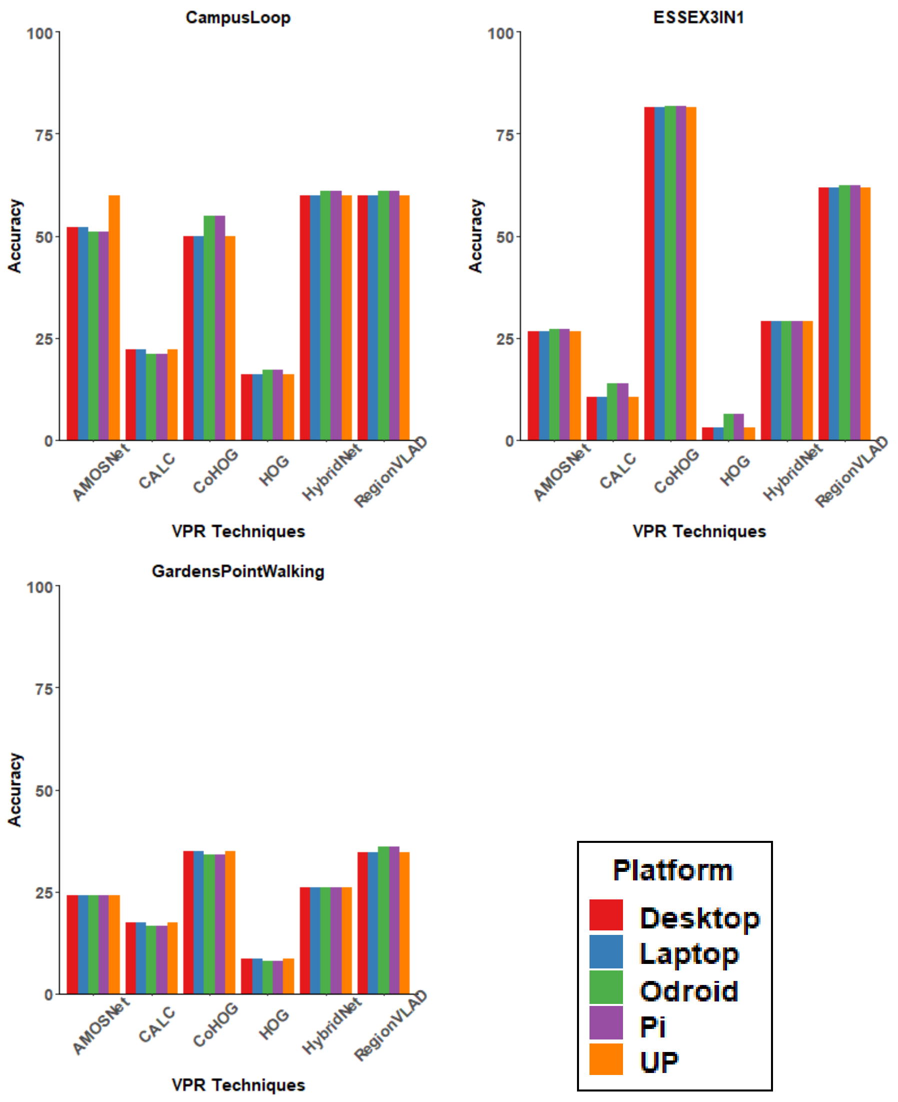

FIGURE 3.1: Graphs showing the accuracy for every VPR technique used in the paper, on each board. The results show that across platform the accuracy remains quite consistent. Note that the ARM platforms produced identical results to each other, as did the x86_64 platforms.

CoHOG

CoHOG [71] uses handcrafted feature-based technique that uses image-entropy to extract regions of interest. HOG [68] [69] descriptors are then assigned to represent each region and the regional descriptors are compared using cosine matching to achieve lateral viewpoint tolerance. One of the inspirations behind this technique was CNN-based techniques' ability to extract regions of interest. CoHOG was developed to achieve state-of-the-art performance without any training requirements, unlike traditional CNN-based techniques. This allows the technique to have a significantly lower feature encoding time.

AMOSNet

AMOSNet was trained on the Specific Places Dataset (SPED) and its deployed model parameters was open sourced by the authors in [76]. The implementation of AMOSNet uses spatial-pyramidal pooling and activations from conv5 layer. Image descriptors are extracted from layer activations and L1-difference is used to match the feature descriptors of the query and reference images.

HybridNet

HybridNet, like AMOSNet, had its model parameters trained on the SPED dataset. However, unlike AMOSNet, the model weights of the top 5 HybridNet convolutional layers are initialised from CaffeNet trained on the ImageNet dataset. The implementation employs a spatial pyramidal pooling on activations from conv5 layer to form feature descriptors. L1 difference is then used to match the query and reference images.

RegionVLAD

Region-VLAD [85] uses a lightweight CNN-based regional approach, as well as VLAD, to overcome the practical deployment limitations of traditional CNNs used for VPR, due to their computational complexity and significant memory overhead. Region-VLAD has shown good image retrieval time for a CNN based method, but still retains a relatively high memory footprint [29]. For the implementation we employ the convolutional layer conv4 of HybridNet, along with 400 region of interests (ROIs). A 256 visual word dictionary is used to extract VLAD descriptors and cosine-similarity is used to match query and reference images.

CALC

CALC [99] was first introduced by Merrill et al. when they trained an autoencoder in an unsupervised manner for the first-time for use in VPR. The objective of the autoencoder was to re-create the HOG descriptors of an image, when given a distorted version the image. Merrill et al. opensourced their implementation using intrinsic AUC computation. For the implementation we use model parameters from the 100K iteration of the auto-encoder on the Places dataset [110].

3.2.3 Computational Platforms

The implementation on each board used the same code libraries and dependencies to ensure consistency across all platforms. However, the libraries used in the implementations were built independently on each platform. This makes the built libraries architecture and processor dependent. This created the architectural differences that are highlighted later in this work, allowing for the evaluation to be commented on, based on the platform architecture.

The chosen platforms for this experiment all utilise CPU computing as opposed to GPU. While the option of using GPU might present several benefits in the way of algorithm efficiency, it was decided instead to use CPU only. This decision was made due to a desire to keep power consumption as low as possible. In the context of the experiment, deploying VPR on UAVs, the need to keep power consumption as low as possible is an important one. The less power consumed, the longer the UAV can perform the task at hand, and the smaller the battery required. This in turn reduces weight and further reduces power consumption, thereby increasing the length of time the UAV can be active for. The code deployed was optimised for CPU use only.

UP-CHT01-A20-0464-A11 [111]

The UP board is marketed as a high performance board with low power consumption. It has a quad-core Atom x5-z8350 processor running at up to 1.92GHz. The UP board is the only board used in this paper that has an Intel processor, like the desktop and laptop standards. The UP board used in the study had 4GB RAM.

TABLE 3.1: CPU usage, memory usage and processing time of RegionVLAD, across all datasets and platforms.

		CPU	
RegionVLAD	Campus	Gardens	ESSEX3IN1
Desktop	34.38	35.50	27.36
Laptop	**100.00**	**99.92**	**99.94**
Odroid	**14.31**	**17.46**	18.47
RPI	21.90	17.49	**17.42**
UP	27.99	28.01	26.79
		Memory	
RegionVLAD	Campus	Gardens	ESSEX3IN1
Desktop	66.25	71.72	79.21
Laptop	69.81	69.14	81.27
Odroid	**53.95**	**51.70**	**51.72**
RPI	**58.90**	**51.00**	**52.55**
UP	70.58	82.86	78.27
		Processing Time	
RegionVLAD	Campus	Gardens	ESSEX3IN1
Desktop	**1.72**	**1.73**	2.45
Laptop	1.76	**1.73**	**2.43**
Odroid	**7.83**	**8.13**	**8.74**
RPI	24.16	33.25	34.31
UP	10.08	10.14	16.31

TABLE 3.2: CPU usage, memory usage and average processing time of CoHOG, across all datasets and platforms.

	CPU		
CoHOG	Campus	Gardens	ESSEX3IN1
Desktop	91.00	96.17	89.20
Laptop	**100.00**	**100.00**	**100.00**
Odroid	**16.08**	**20.32**	**20.47**
RPI	25.84	25.74	25.74
UP	79.29	86.82	79.48
	Memory		
CoHOG	Campus	Gardens	ESSEX3IN1
Desktop	64.61	70.53	78.71
Laptop	68.40	67.72	82.65
Odroid	**45.85**	**46.48**	**53.29**
RPI	**49.16**	**61.26**	**66.95**
UP	68.02	80.72	78.68
	Processing Time		
CoHOG	Campus	Gardens	ESSEX3IN1
Desktop	0.36	0.58	0.57
Laptop	**0.24**	**0.34**	**0.39**
Odroid	8.49	18.44	17.20
RPI	5.38	10.03	10.51
UP	**1.24**	**1.96**	**2.36**

TABLE 3.3: CPU usage, memory usage and average processing time of CALC, across all datasets and platforms.

	CPU		
CALC	Campus	Gardens	ESSEX3IN1
Desktop	100.14	100.14	87.07
Laptop	100.00	100.00	100.00
Odroid	**22.70**	**23.27**	**21.35**
RPI	28.89	29.01	29.58
UP	27.57	27.71	29.30
	Memory		
CALC	Campus	Gardens	ESSEX3IN1
Desktop	64.95	70.34	81.67
Laptop	67.87	66.64	84.84
Odroid	**45.15**	**48.31**	**56.47**
RPI	**55.27**	**66.08**	**70.01**
UP	68.05	80.57	78.40
	Processing Time		
CALC	Campus	Gardens	ESSEX3IN1
Desktop	**0.07**	**0.07**	**0.12**
Laptop	0.17	0.14	0.21
Odroid	0.68	0.69	0.70
RPI	**0.56**	**0.55**	**0.61**
UP	0.67	0.68	1.04

TABLE 3.4: CPU usage, memory usage and average processing time of HybridNet, across all datasets and platforms.

	CPU		
HybridNet	Campus	Gardens	ESSEX3IN1
Desktop	54.10	57.08	40.07
Laptop	**100.00**	**100.00**	**100.00**
Odroid	**18.83**	**20.80**	**21.36**
RPI	25.66	25.69	25.78
UP	25.95	25.97	25.57
	Memory		
HybridNet	Campus	Gardens	ESSEX3IN1
Desktop	67.69	71.16	83.02
Laptop	69.06	67.86	85.55
Odroid	**55.35**	**57.17**	**56.30**
RPI	68.40	68.48	71.37
UP	70.94	82.85	78.49
	Processing Time		
HybridNet	Campus	Gardens	ESSEX3IN1
Desktop	**1.18**	**1.17**	**1.85**
Laptop	4.81	4.80	5.54
Odroid	25.56	26.21	27.21
RPI	**20.72**	**20.65**	**20.88**
UP	24.57	24.47	31.38

TABLE 3.5: CPU usage, memory usage and average processing time of HOG, across all datasets and platforms.

	CPU		
HOG	Campus	Gardens	ESSEX3IN1
Desktop	41.50	27.40	22.42
Laptop	**78.67**	**100.00**	**100.00**
Odroid	**22.45**	**24.94**	**23.61**
RPI	29.36	29.00	28.98
UP	31.70	31.91	26.11
	Memory		
HOG	Campus	Gardens	ESSEX3IN1
Desktop	68.13	70.31	82.88
Laptop	67.08	66.43	85.11
Odroid	**40.30**	**44.25**	**43.34**
RPI	**63.38**	**56.41**	**57.96**
UP	75.74	78.82	66.33
	Processing Time		
HOG	Campus	Gardens	ESSEX3IN1
Desktop	**0.02**	**0.02**	0.09
Laptop	**0.02**	0.03	**0.07**
Odroid	0.10	0.15	**0.17**
RPI	0.20	0.27	0.30
UP	**0.09**	**0.11**	0.40

TABLE 3.6: CPU usage, memory usage and average processing time of AMOSNet, across all datasets and platforms.

	CPU		
AMOSNet	Campus	Garden	ESSEX3IN1
Desktop	54.22	57.24	40.02
Laptop	**100.00**	**100.00**	**100.00**
Odroid	**17.14**	**20.79**	**21.03**
RPI	25.64	25.78	25.71
UP	25.95	25.94	25.80
	Memory		
AMOSNet	Campus	Garden	ESSEX3IN1
Desktop	69.06	71.15	83.52
Laptop	69.27	67.98	85.64
Odroid	**51.28**	**54.75**	**53.99**
RPI	76.11	69.88	68.44
UP	77.35	80.20	78.94
	Processing Times		
AMOSNet	Campus	Garden	ESSEX3IN1
Desktop	**1.17**	**1.16**	**1.84**
Laptop	4.73	4.84	5.61
Odroid	25.56	26.76	26.90
RPI	**20.64**	**20.51**	**20.80**
UP	24.08	24.48	30.21

Raspberry Pi 3 Model B [112]

The Raspberry Pi was developed by the Raspberry Pi Foundation with the desire to create a computer that was not only cheap but could also be considered as disposable. The Raspberry Pi used in this paper is the earliest model of the thrid-generation of Raspberry Pi and is the replacement to the Raspberry Pi 2 Model B. This model has a quad-core ARM processor with arm64 architecture.

The Raspberry Pi used in the setup had 1GB RAM and 100Mb of swap. For the setup we increased the swap space to 2GB. This was necessary to keep up with the computational demands of the various VPR techniques used in the experiment. It should be noted however, that increasing the size of the swap greatly increases the time taken for computation.

ODROID XU4 [113]

The Odroid board used in the study had 2GB RAM and no swap as default. However for the setup we created 2GB of swap. Similarly to the Raspberry Pi, this was necessary to keep up with the computational demands of the VPR techniques used throughout the experiment. The Odroid board is also the only board used for the study presented in this paper that has a fan. The fan fires periodically causing a surge in power consumption. While it is possible to disconnect the fan from the board, for the sake of not damaging the board during the course of these experiments, the fan was kept attached.

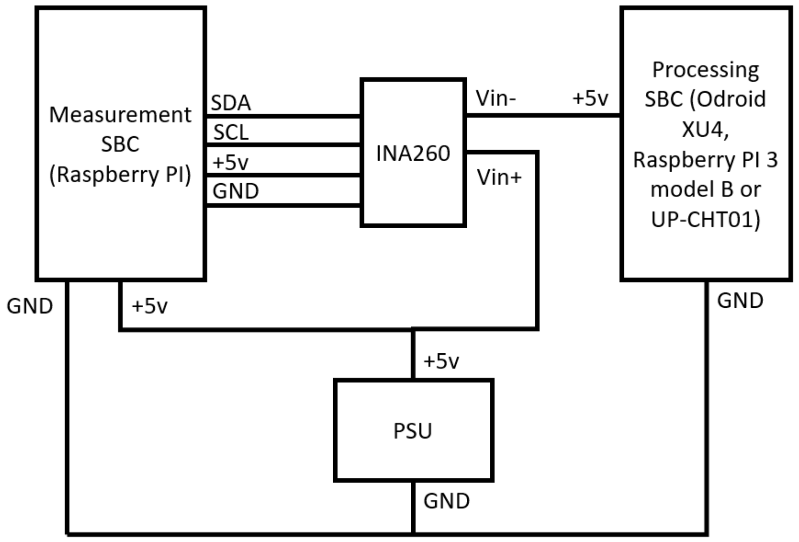
Measurement SBC (Raspberry PI)
SDA
SCL
+5v
GND
INA260
Vin-
Vin+
+5v
Processing SBC (Odroid XU4, Raspberry PI 3 model B or UP-CHT01)
GND
+5v
GND
+5v
PSU
GND

FIGURE 3.2: Power measurement circuit schematic.

Laptop

This platform is a DELL XPS 13 9380. It has an Intel(R) Core i7-8565U CPU, operating at 1.99GHz, and 16GB of RAM. Due to its large size, weight and power requirements, it is not feasible to use as an embedded platform. As such, it is only included as a reference for the other embedded boards

Desktop

As well as the laptop, a desktop computer was included as an additional reference platform. This desktop had an AMD RYZEN 1400 Quad-Core Processor, operating at 3.20 GHz, and 32GB of RAM.

3.2.4 Evaluation Metrics

Matching Performance

The image with the highest matching score extracted from the mapped reference dataset should correspond to the query image of the same location. As such, a match is determined as correct if, given a query image, the technique assigns the corresponding reference image the highest similarity score, determined for each technique, of every possible image. Determining the percentage of correctly matched images for each of the datasets used will give the accuracy of each VPR technique.

Algorithm Efficiency

In this paper the efficiency of each technique used is evaluated by measuring the CPU usage, memory usage and processing time of each technique. CPU usage and memory usage are reported as a percentage. Memory usage is measured as the percentage of active memory out of total memory. In addition, processing time is recorded, reported in seconds. This includes the feature encoding time for an input query image, as well as the descriptor matching time for the number of reference images in the dataset, image loading and preprocessing. In order to benchmark the

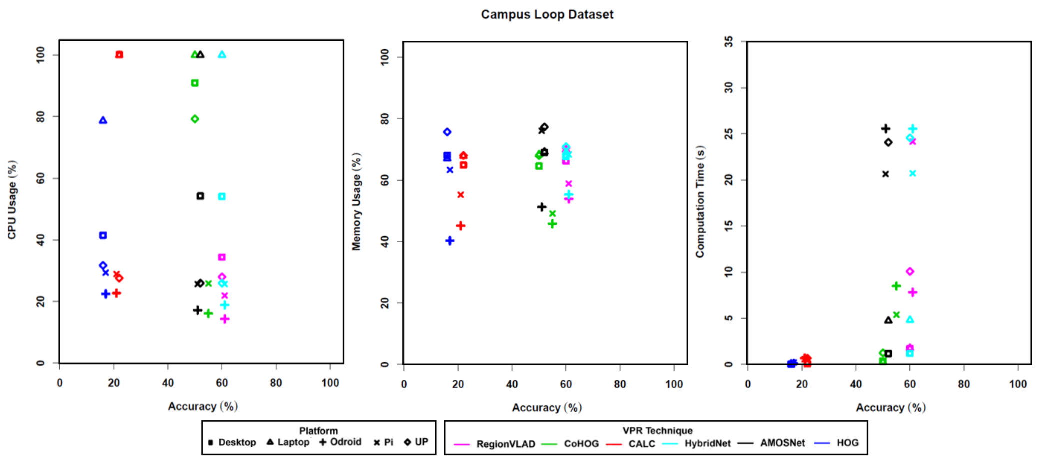

FIGURE 3.3: Graphs showing the CPU usage, memory usage and processing time, for every VPR technique used in the paper. The processing time was reported as the average encoding and matching time of all query images in the Campus Loop dataset (100 images). Each VPR technique has a unique colour and each platform is represented by a unique shape.

boards accurately the images were loaded from a dataset. In real-world applications, the images would be acquired from a camera in real-time.

Power Consumption

In addition to evaluating the precision-recall of the techniques produced by each board, the power consumption of each board was measured using a setup of a INA260 module [114] monitored by an additional Raspberry Pi, the configuration of which is shown in figure 3.2. The power consumption of each embedded platform was recorded whilst each platform first encoded a query image and then attempted to match that image to its already complied map. The power consumption was continuously measured as every query image in the dataset was processed and matched sequentially. This gave an indication of the power consumption over the course of encoding and matching images.

Weight

Another interesting factor to take into consideration is weight. In the context of UAVs, the weight of an embedded board will affect the overall power consumption and therefore flight time of the system. A heavier board or processor will cause a higher power consumption meaning that the possible time of flight will be reduced. This paper takes into consideration the weights of the embedded boards used in these experiments.

It should be noted that as this is a lab experiment, the boards evaluated are standard production models. Production robots may use custom boards that have been refined for the target application, potentially reducing weight and power consumption.

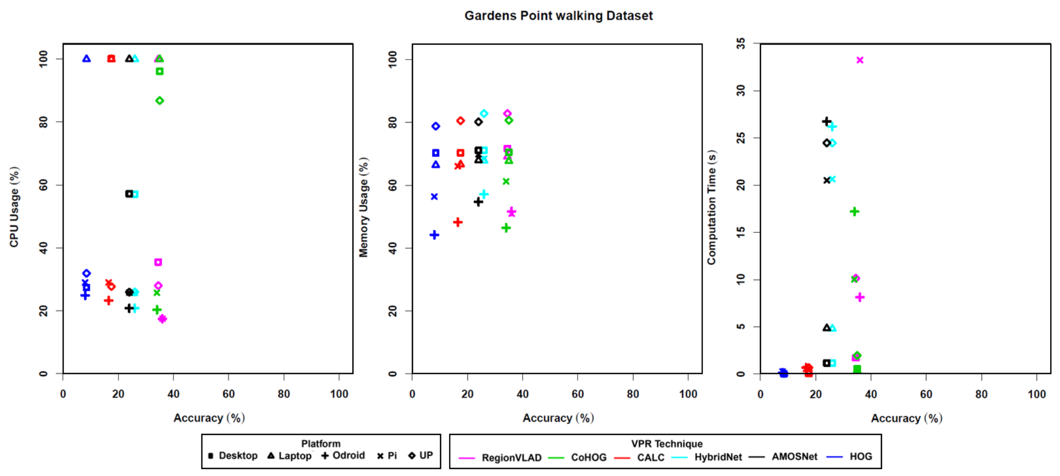

FIGURE 3.4: Graphs showing the CPU usage, memory usage and processing time, for every VPR technique used in the paper. The processing time was reported as the average encoding and matching time of all query images in the GardensPointWalking dataset (200 images). Each VPR technique has a unique colour and each platform is represented by a unique shape.

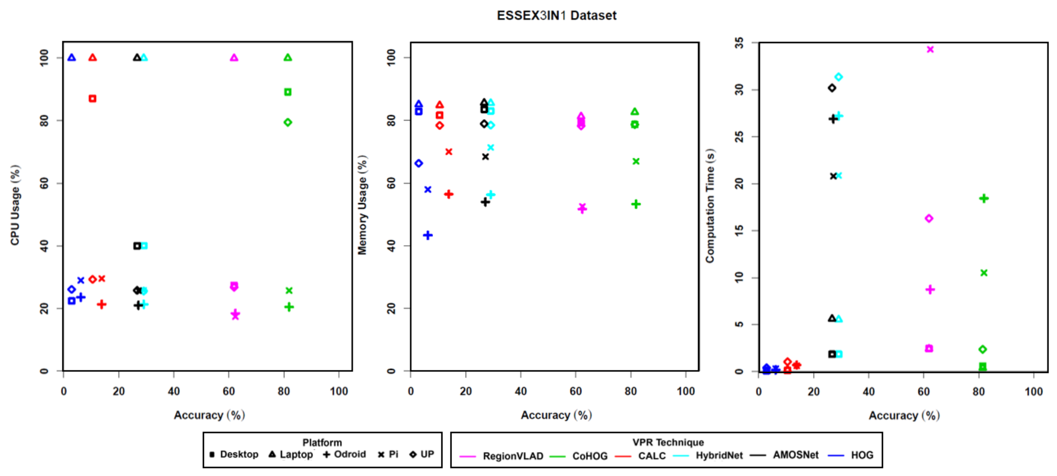

FIGURE 3.5: Graphs showing the CPU usage, memory usage and processing time, for every VPR technique used in the paper. The processing time was reported as the average encoding and matching time of all query images in the ESSEX3IN1 dataset (210 images). Each VPR technique has a unique colour and each platform is represented by a unique shape.

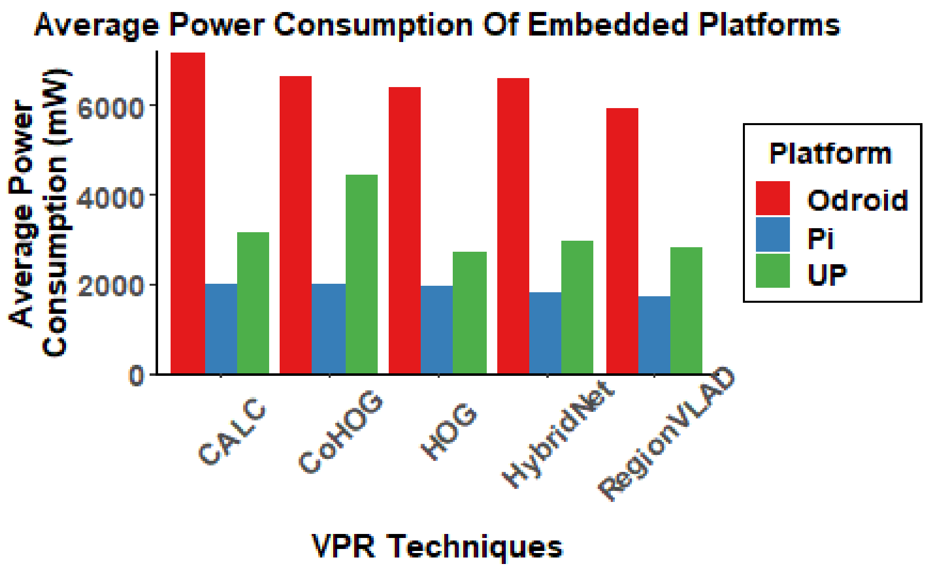

FIGURE 3.6: Average power consumption of each VPR technique when implemented on the embedded platforms.

3.3 Results and Analysis

This section contains results based on all evaluation metrics for all computational platforms.

3.3.1 Accuracy

This sub-section reports the findings of the matching performance of each technique on all three datasets used for testing as shown in figure 3.1.

As part of this thesis, it was discovered that the Raspberry Pi and Odroid produced the same results in terms of accuracy for each VPR technique. Conversely, the UP board, laptop and desktop platforms all produced the same results in terms of accuracy as well. This is likely due to the fact that both the Odroid and Raspberry Pi platforms use ARM processors, having arm64 or armhf architecture, and the UP board, laptop and desktop use Intel Atom, Intel i7 and AMD Ryzen 5 processors respectively, all sharing the x86_64 architecture. Ubuntu 18 operating system was used across all platforms. Apart from architecture type, the only difference in OS is that the raspberry Pi uses a 32-bit OS, however it produced they same accuracy as the Odroid board which used a 64-bit OS and had the same ARM architecture suggesting that the difference in 32-bit, 64-bit systems does not have a significant affect on the accuracy of the techniques. The running environments were kept the same for each platform using the same version of Python and additional libraries, with the only difference being when a specific version had to be built for ARM architecture as opposed to the standard x86_64 architecture.

3.3.2 Algorithm Efficiency

This sub-section reports the findings of the algorithm efficiency of each platform and technique as performed on all three datasets used for testing.

CPU Usage

In terms of CPU usage each platform and technique combination only displayed slight variation between datasets as shown in tables 3.1, 3.2, 3.3, 3.4, 3.5 and 3.6.

The laptop displayed the highest CPU utilisation by far, using roughly 100% of the CPU's capacity for all techniques. In comparison, the Ordoid and Raspberry Pi platforms had very low CPU utilisation for all techniques, using between 15-30% of their respective CPU's capacity. In addition, the UP board has relatively low CPU utilisation for all techniques except for CoHOG, which uses an average of 80% of its CPU's capacity to perform. This is likely due to an implimentation characteristic of x86_64 architecture that is favourable to the CoHOG tecnique, as UP is the only x86_64 embedded platform. The desktop also displayed a high CPU usage when running CoHOG and CALC, but had a relatively low CPU usage for the remaining techniques.

The embedded platforms demonstrated low CPU utilisation at times. This is likely due to their limited memory and consequent need to page to relatively slow SD card storage. This was not exhibited by the non-embedded platforms, that had enough memory to not require paging. Whilst the need to page caused a slight increase in the computational time, it suggests that potentially a more computationally complex algorithm could be run on the embedded platforms with similar success, given a larger memory capacity.

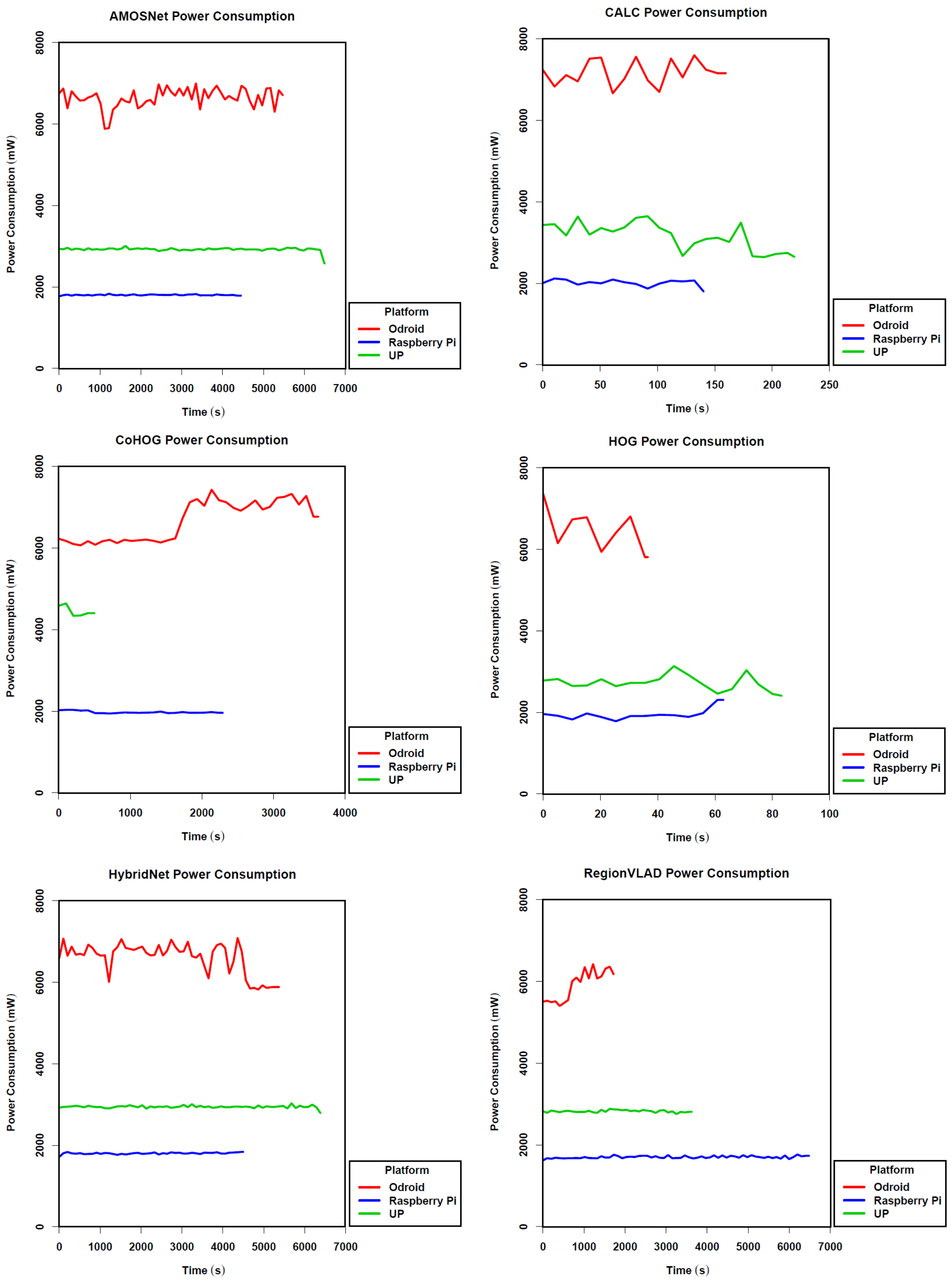

FIGURE 3.7: Graphs showing the power consumption over time on each board, for every VPR technique used in the paper. The power consumption was measured during the encoding and matching of every query image in the ESSEX3IN1 dataset (210 images).

TABLE 3.7: Table showing specifications for each board.

Name	Processor	Architecture	Memory
UP	Intel Atom x5-Z8350 Quad-Core Processor 1.92GHz	x86_64	4GB
Raspberry Pi 3 model B	Quad Core 1.2GHz Broadcom BCM2837 64bit CPU	arm64	1GB
Odroid XU4	Samsung Exynos5422 ARM Cortex-A15 Quad 2.0GHz	armhf	2GB
Laptop	Intel(R) Core i7-8565U CPU 1.8GHz 1.99GHz	x86_64	16GB
Desktop	AMD RYZEN 1400 Quad-Core Processor 3.20 GHz	x86_64	32GB

Memory Usage

Trends in memory utilization are difficult to identify due to variation in image resolution between datasets, the amount of memory each board has and the technique used, as shown by the results in tables 3.1, 3.2, 3.3, 3.4, 3.5 and 3.6. During the experiments in this paper the images were loaded one at a time for processing to cut down on memory consumption given the limited memory capacity of the embedded platforms. As such, no more than one image from the dataset is stored in memory at any given time. Memory utilisation appears also to be affected by platform architecture.

The ARM architecture platforms utilised the least memory, however this is likely due to the resized and compressed nature of the datasets, required to make the algorithms run on those platforms. After further experiments, it is estimated that the memory utilisation of the x86 platforms could be up to 30% lower if given the same compressed images.

Processing Time

As expected, the laptop and desktop had a much shorter processing time than any of the embedded platforms, in some cases up to 3 times faster than the fastest of the embedded platforms. In general, all platforms and techniques took slightly longer when tested with the ESSEX3IN1 dataset, as shown in tables 3.1, 3.2, 3.3, 3.4, 3.5 and 3.6, likely due to the uninformative images in this dataset. However, the results are similar across all datasets.

The Raspberry Pi took the longest on all three datasets to complete RegionVLAD, taking over double the time of the next platform. However, in contrast, the Pi had the shortest processing time for HybridNet across all datasets. Despite this the processing time for HybridNet was significantly longer than the other techniques regardless of platform. UP had by far the shortest processing time using CoHOG, greatly outperforming the other embedded platforms in terms of processing time.

All platforms perform well using both CALC and HOG with the shortest processing times of all the techniques.

3.3.3 Power Consumption

This sub-section reports the findings of the power consumption of each technique on all three of the embedded platforms used for testing as shown in figure 3.6.

The Raspberry Pi showed a clear superiority with both low power consumption and relatively low computation time despite being constrained by RAM limitations and having to use swap space on the micro-SD card. This low power consumption is likely due to having an ARM processor and no active cooling unit. In comparison the Odroid board consistently demonstrated the highest power consumption per unit time of the three embedded platforms at least partially due to its active cooling unit. This comparatively high-power consumption makes for an unfavourable trade-off between computation time and power consumption. The Odroid board

TABLE 3.8: Table showing weights for each board.

Name	Weight
UP	97
Raspberry Pi 3 model B	43
Odroid XU4	64
Laptop	1221
Desktop	N/A

also suffered from RAM limitations, resulting in paging to the micro-SD. The UP board had a relatively low power consumption per unit time but took the longest on most techniques. Despite taking the longest time for most methods, this is not significant, giving the board an acceptable trade-off between computation time and power consumption. Despite the slightly longer computation time, the UP board was less constrained by RAM than the other embedded platforms.

3.3.4 Weight

Another factor to take into consideration is the weight of each embedded board, as noted in Table 3.8. A heavier board will require the UAV carrying it to consume more power to fly. Therefore, a lighter board is usually preferable. However, if this lighter board comes at the expense of a board that draws more power for computation, this is not as favourable and another factor is added to the trade-off between computation time and power consumption.

Of the boards, the Raspberry Pi is the lightest, at just over 2/3 the weight of the next lightest board. This coupled with the lowest power consumption for processing of the three boards makes it highly recommended. In contrast, the UP board is the heaviest, at over double the weight of the Pi. This is due to its passive cooling unit which makes the trade-off not as favourable. Finally, the Odroid board occupies the middle ground between the other two in terms of weight. However, as stated before this lighter board comes at the expensive of a high-power consumption, again making the trade-off unfavourable. None of the embedded boards showed an advantage over the others in terms of RMF. Therefore, power consumption and weight are the primary factors when selecting a board.

3.4 Summary

As expected, the overall performance of the high-end platforms was greater than those of the embedded platforms. However, the embedded platforms' ability to match the accuracy of the high-end platforms indicates good potential for future use in the field of VPR. Out of the three embedded platforms assessed the Raspberry Pi showed clear superiority, with the lowest power consumption, good processing time and similar accuracy to the high-end platforms. Apart from the UP platform, the embedded platforms struggled with descriptor size until additional swap space was added, when performance was acceptable. Descriptor size will become less of a problem as embedded platform's memory capacity increases.

In terms of power consumption, the Raspberry Pi and Odroid platforms showed the same trends in power consumption per technique, with the most power consumed when using CALC and the least when using RegionVLAD. On the other hand, when the techniques were run on the UP board, CoHOG had the highest power consumption. This is likely due to the UP board utilizing up to 4 cores when performing CoHOG, whereas the Odroid and Pi did not use more than 2 when running the same algorithm. This is potentially due to architectural differences between the x86_64 architecture of the UP board and the ARM architecture of the Odroid and Pi.

Chapter 4

Real-Time Matched Frames

Benchmarking has a significant impact on the understanding of a model's useability within a target application. Thus, evaluating VPR techniques is an important area of computer vision research. This thesis proposes an evaluation framework intended to provide useful information about a VPR technique's usability within a real-time application. This approach makes use of the incoming frame rate of an image stream and the VPR frame rate, the rate at which the technique can perform VPR, to determine how efficient various VPR techniques would be at performing VPR in real-time. This is important because it provides new insight into the use and practicality of VPR in a real-time environment.

4.1 Background

Autonomous operation of a mobile robotic platform requires the ability to locate and identify the platform's own location within an environment. Simultaneous localisation and mapping (SLAM) [31] is a widely researched area of autonomous robotics that would enable a mobile robot to self-localise within an environment and maintain an accurate map of the surrounding environment. Typically, robots in these scenarios are equipped with a wide variety of sensors to provide essential location and motion information. In these situations, dead-reckoning is used to estimate the robots position using collected sensor information. However, consecutive dead-reckoning estimations accumulate errors, which become more significant the longer the robot's trajectories are. This causes incorrect assumptions of the robot's location within the environment. These accumulated errors can be corrected if the robot revisits and recognises a previously visited location within the environment. This is known as 'loop-closure'. In a vision-based system, this is achieved if a robot is able to recognise a previously visited location using only visual cues. Recalling a previously visited location using only visual cues, deemed Visual Place Recognition (VPR), has become a subject of great interest within the robotic vision community.

Over the years, many techniques have been developed to tackle the exceedingly difficult task of VPR. A technique must be able to correctly identify whether, given an image, the location has been visited before, as well as be able to correctly identify and handle false positive results. Additionally, in order to be useful in autonomous navigation, a VPR technique must be able to run, and handle image processing and recognition, in real-time. Due to the number of different VPR techniques that have been developed, each with their own strengths and weaknesses, it is useful to be able to compare these techniques.

While there are many evaluation metrics that are widely used to evaluate Visual Place Recognition (VPR) techniques, such as AUC-PR, RecallRate@N and EP, which try to quantitatively summarise the matching performance of a VPR technique, their extension to real-world scenarios can be at times ambiguous. The results obtained by these evaluation metrics are obtained based on the processing of an algorithm which is fed images from a dataset one at a time when the previous image has finished processing. Subsequently, although higher AUC-PR/EP/RecallRate may reflect that a technique retrieves mostly correct matches, real-world factors like image-retrieval time and platform-speed are neglected. This means a highly compute-intensive technique that cannot be used in a real-world application (or in real-time) may be ranked above others.

In a real-time scenario, cameras can provide up to 50 images per second. While this is useful, if the VPR algorithm implemented cannot process images as fast as a camera can provide them, then a decision must be made between processing the next image provided or the image provided at the time. As such an algorithm with high accuracy but also high computational time may be less effective in a real-time scenario than an algorithm with low accuracy and low computational time.

This section provides a benchmark allowing for the evaluation of algorithms with a view to real-world, real-time VPR. This work is not intended as a discreditation to already established benchmarking methods. Instead it is intended to provide additional methods of evaluation specifically geared towards real-time evaluation and approximation. The use of real-time matched frames as a benchmark has no bearing on VPR not performed in real-time.

4.2 Methodology

The following benchmark is aimed to allow for the evaluation of VPR algorithms with a view to real-world, real-time applications. The proposal of this benchmark is that the critical ratio of Incoming frame rate and VPR frame Rate will determine whether a VPR algorithm will perform well in a real-world environment.

Let the sampling rate of the camera, which is usually a fixed value denoted as frames per second (FPS), be F. Let the frames sampled per distance be D and the speed of the platform be V. The incoming frame rate from the platforms camera, or other vision sensor, can be computed as:

$$Incoming\ Frame\ Rate = min(K \times D \times V, F) \tag{4.1}$$

In this equation K represents a unit-less constant that represents further down-sampling cause by the vision pipeline. The VPR frame rate, which can be described as the potential VPR matches computed per second, can be calculated as shown in equation 4.2.

$$VPR\ Frame\ Rate = Floor(\frac{1}{t_R}) \tag{4.2}$$

The ratio G, as shown in equation 4.3, can have a maximum possible value of 1. This is due to the fact that the maximum number of VPR query images that can be matched can only be equal to the total incoming images provided.

$$G = Floor(max(\frac{Incoming\ Frame\ Rate}{VPR\ Frame\ Rate})) \tag{4.3}$$

It should be noted that the modelling of Incoming Frame Rate is one traversal-specific and that the Incoming Frame Rate can also be an explicit specification from either the computer vision or a robotics VPR application.

Let's say that on a particular dataset traversal, all of the query images have been matched by a VPR technique based on the ground-truth. A Boolean list $L_{matches}$ is

Algorithm 1 Computing RMF

```
Original_Matches_List = $List_{matches}$
$N_q = Length(List_{matches})$
$M_q = Sum(List_{matches})$
$RMF = 0$
$V, D, F, K : Given$
$G : Computed$
for all $index, element\ in\ List_{matches}$ do
    if $(index + 1)\%G = 0$ or $index = 0$ then
        if $element = 1$ then
            $RMF = RMF + 1$
        end if
    end if
end for
return $M_q, RMF$
```

calculated based on all query frames in order of traversal, where the list element 1 identifies a true positive and the list element 2 identifies a false-positive result. Often within VPR this information is already computed for plotting PR-Curves. Given the ratio G, a VPR technique will only be able to perform VPR for a query frame after some constant frame interval based on G.

The total correctly matched query images in a VPR dataset at the maximum value of recall can be denoted as M_q out of a total of N_q query images in the dataset. Based on the prospective loss of potential place matching candidates for a slow VPR technique due to $G > 1$, the number of total matched query images will reduce to Real-time Matched Frames (RMF) out of a total of N_q query images. This RMF can be interpreted as an active evaluation metric and we compute this RMF using Algorithm 1. With all the parameters required to calculate RMF combined, it can be thought of as a simulation of a traversal through an environment at a given speed, reference map and sampling rate.

4.3 Experimental Setup

4.3.1 Evaluation Datasets

As stated in Chapters 1 and 3, there have been many datasets developed specifically for the task of VPR. As part of the experiment laid out in this Chapter we chose several datasets for assessing each technique against the evaluation criteria. The datasets chosen were the same a described in Chapter 3.

4.3.2 VPR Techniques

The experiment in this chapter is carried out on a number of state-of-the-art VPR techniques. The techniques used are the same as described in Chapter 3. The selected techniques were chosen to include a range of VPR technique methodologies. HOG and CoHOG are traditional handcrafted feature-based techniques, they require no training before use. AMOSNet, HybridNet and RegionVLAD are CNN-based techniques, with RegionVLAD including an additional pooling stage to compute image descriptors. CALC is based on the use of an auto-encoder.

4.3.3 Evaluation Metrics

In addition to the evaluation metric presented in this chapter, the matching performance of each technique on each platform is evaluated to further demonstrate the need for an evaluation metric designed with a view to real-time, real-world applications.

Matching Performance

The image with the highest matching score extracted from the mapped reference dataset should correspond to the query image of the same location. Each match and its corresponding similarity score are then used to calculate the precision-recall value for each technique, for each dataset, for each platform.

The area under a precision-recall curve (AUC) is widely accepted as a good method of evaluating the matching performance of a VPR technique. For our study, we compute the precision and recall for every matched and unmatched query image. Using these values, we compute the AUC using the equation 4.4.

$$\begin{aligned} AUC = \sum_{i=1}^{N-1} \frac{p_i + p_{i+1}}{2} \times (r_{i+1} - r_i) \\ where;\ N &= No.\,of\ Query\ Images \\ p_i &= Precision\ at\ i \\ r_i &= Recall\ at\ i \end{aligned} \tag{4.4}$$

4.4 Results and Analysis

4.4.1 Real-Time Matched Frames (RMF)

Overall, CALC managed to successfully match the most frames across all platforms and datasets, as presented by the graphs in figure 4.1, often greatly out-performing the other techniques. Out of all platforms, the laptop demonstrated the highest number of successfully matched frames within the real-time constraints, significantly more than the UP or Odroid boards despite having comparable processor speeds. The desktop platform had the highest processor speed, despite this it has the second highest number of matched frames. The Raspberry Pi had the least overall number of correctly matched frames. This makes sense as the Raspberry Pi has the lowest processor speed of all platforms.

Campus Loop

Across all platforms, CALC correctly matched the most frames within the real-time constraints greatly out-performing all the other techniques. RegionVLAD and Co-HOG perform relatively consistently across all platforms, albeit quite poorly. The remaining techniques also perform relatively poorly on the Desktop and Laptop platforms however their performance drops when performed on the embedded platforms.

ESSEX3IN1

Similarly, to the campus dataset, CALC performs the best on the Desktop, Laptop and Odroid platforms, however on the remaining two platforms CALC has a similar performance as all the other VPR techniques. The other techniques all perform similarly poorly across all platforms.

GardensPointWalking

Similar to the other two datasets, CALC again performs the best on all platforms, except for the Raspberry Pi. Aside from RegionVLAD, which demonstrates a relatively consistent performance, the remaining techniques struggle greatly on the embedded platforms, unable to successfully match any frames within the real-time constraints.

4.4.2 Performance Benefit Curve

This sub-section reports the findings of the Precision-Recall of each technique on all three datasets used for testing as shown in figures 4.2, 4.3, 4.5, 4.4 and 4.6.

As part of the findings of the experiment, it was discovered that the Raspberry Pi and Odroid produced the same results in terms of Precision-recall. Similarly, the UP board, laptop and desktop platforms all produced the same results in terms of precision-recall. This is likely due to fact that both the Odroid and Raspberry Pi boards use ARM processors, having arm64 or armhf architecture, and the UP board, laptop and desktop use Intel Atom, Intel i7 and AMD Ryzen 5 processors respectively, all sharing the x86_64 architecture.

ESSEX3IN1 Dataset

The ARM based platforms achieved the highest results using RegionVLAD, CoHOG and HOG on the ESSEX3IN1 dataset, compared to the Intel/AMD platforms. In addition, the raspberry Pi and Odroid achieved the highest results using CALC, greatly out-performing the remaining platforms. Conversely, the Raspberry Pi and Odroid boards achieve a slightly lower performance than the other platforms when using AMOSNet. When using HybridNet, all platforms performed well, with only slight variation between them.

Campus Loop

All platforms performed well on this dataset using RegionVLAD, HybridNet and AMOSNet, with only slight variation between them, achieving state-of-the-art performance using RegionVLAD. For the other techniques, the UP board, laptop and desktop all performed well using CoHOG, with Raspberry Pi and Odroid achieving only a slightly lower performance. In common with the ESSEX3IN1 dataset, the Raspberry Pi and Odroid achieved the highest results using CALC, slightly out-performing the remaining platforms. In addition, the Raspberry Pi and Odroid boards achieved the highest results using HOG for this dataset, greatly out-performing the remaining platforms.

GardensPointWalking

The Raspberry Pi and Odroid achieved the highest results when using RegionVLAD and CoHOG on this dataset, closely followed by that of the UP board, laptop and

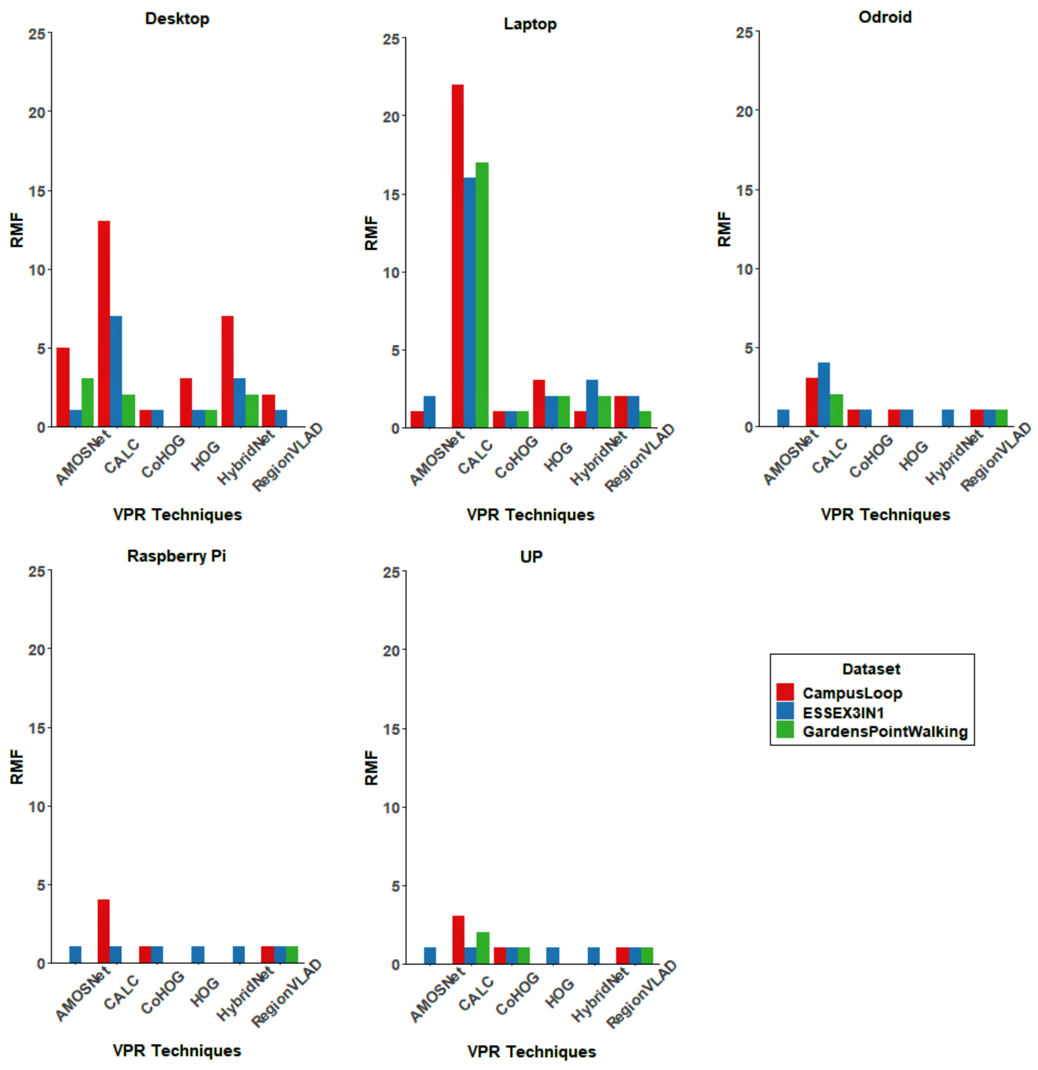

FIGURE 4.1: Graphs showing the Real-Time Matched Frames (RMF) for every VPR technique used in the paper, on each board. CALC achieved the highest RMF across all platforms, greatly outperforming the other techniques. Of the platforms, the laptop demonstrated the highest number of successfully matched frames within the real-time constraints.

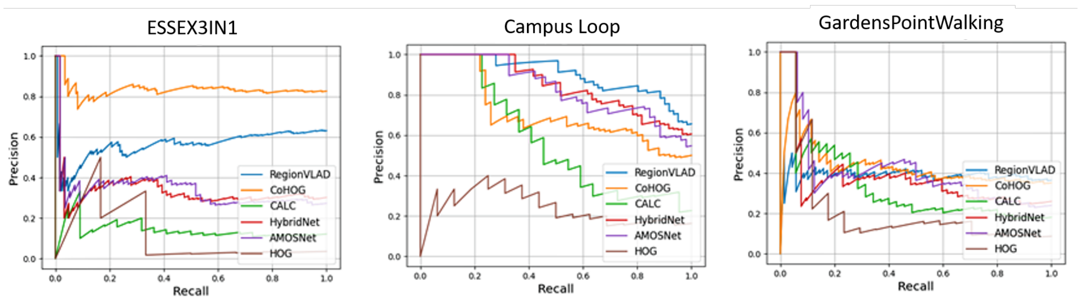

FIGURE 4.2: Graphs showing the precision-recall of every VPR technique and dataset on the desktop platform.

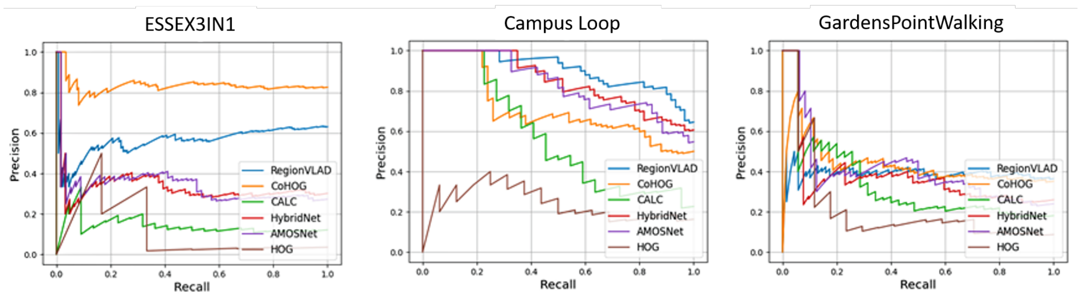

FIGURE 4.3: Graphs showing the precision-recall of every VPR technique and dataset on the laptop platform.

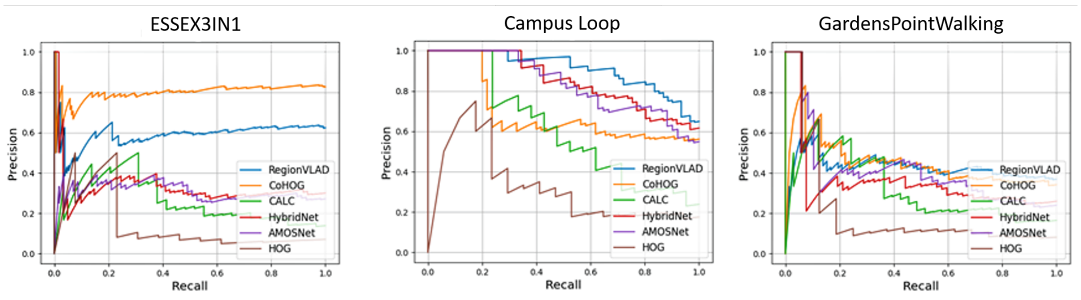

FIGURE 4.4: Graphs showing the precision-recall of every VPR technique and dataset on the Raspberry Pi platform.

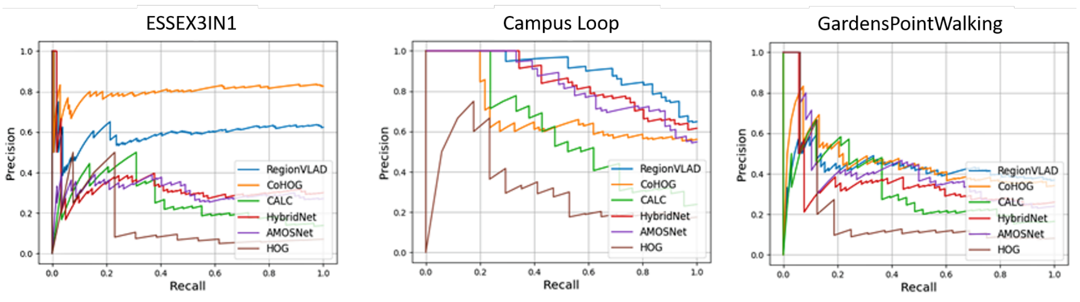

FIGURE 4.5: Graphs showing the precision-recall of every VPR technique and dataset on the odroid platform.

desktop. The UP board, laptop and desktop all performed well using CALC, HybridNet and HOG, with Raspberry Pi and Odroid achieving only a slightly lower performance. All platforms achieved the same performance using AMOSNet for this dataset.

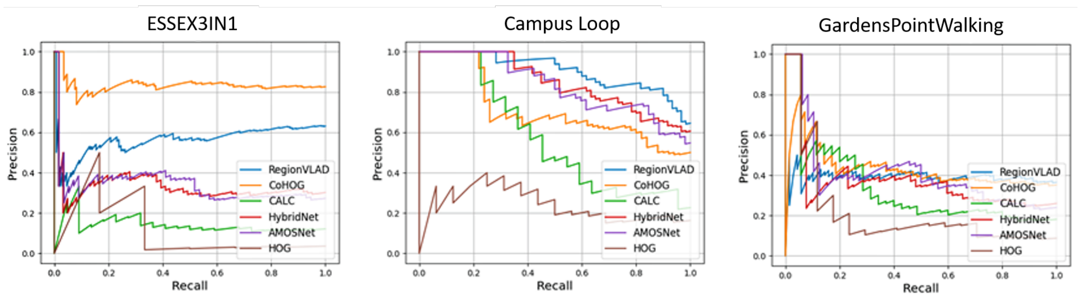

FIGURE 4.6: Graphs showing the precision-recall of every VPR technique and dataset on the UP platform.

4.5 Summary

The results of the RMF experiment clearly favoured CALC as the VPR technique with the best real-world, real-time applicability. On the contrary, in the results of the more traditional PR-curve evaluation, CALC had one of the lowest performances. This indicates a trade-off between precision and computation time. Within this trade-off it can be seen that CALC is superior when attempting to maximise performance in terms of both accuracy and time across all platforms.

A clear limitation of RMF is in the assumption that the platform in question if always moving at a constant velocity. However, in a real-world scenario, this is not likely to be the case. For the purpose of benchmarking, this limitation can be largely mitigated by partitioning the dataset sequence into sub-sequences that contain near-constant velocities and processing them individually.

It can also be noted that due to the nature of its calculation, RMF will inherently favour techniques that have true positive results distributed fairly evenly throughout a traversal. This is opposed to techniques that have their true positives results concentrated in certain regions of a traversal. This is due to the distance-based sampling approach of RMF. A better analysis could be obtained by combining RMF with the true-positives (loop-closure) distribution over a dataset trajectory, as proposed by Porav et al. [115]. The metric proposed in [115] compliments the value proposed by RMF for applications that require a loop-closure every few meters, otherwise the localisation drift error becomes too large to handle.

Chapter 5

Conclusion and Future Work

Firstly, this section will conclude the findings of the research discussed within this report and results of the experimentation undertaken throughout the course of this masters. Secondly, the potential directions of work continuing from this research will be discussed.

5.1 Conclusion

Overall, the setup and results of two experiments were presented in this dissertation. As well as some conclusions that could be drawn from the results and the importance and value of the contributions presented.

The first experiment, as presented in chapter 3 focused on benchmarking the performance of VPR techniques deployed on embedded platforms. The objective being to understand what effects, if any, the architecture and build of an embedded platform has on the accuracy and efficiency of each technique.

The high-end platforms, the reference laptop and desktop, achieved a better performance across all techniques in comparison to the other platforms, in terms of the efficiency metrics like CPU and Memory utilization. In retrospect this was to be expected as these platforms possess far superior processing and memory capabilities. However, it can be noted that the embedded platforms; odroid, raspberry Pi and UP, all achieved a similar accuracy to the high-end platforms. This indicated that the embedded platforms have good potential for use in VPR applications.

Of the embedded platforms, the Raspberry Pi was shown to have superior power consumption and good processing time, making it the best suited to VPR of the three platforms. It was noted that the embedded often struggled with the size of the descriptors generated by each technique until additional swap space was added. However, this will become less of a problem in the future as platforms are improved and subsequent memory capacity increases.

As a continuation of the first experiment, the second experiment, as presented in chapter 4 sort to investigate the potential of VPR techniques in a real-time application. This experiment, like the first, was performed on several platforms in order to investigate whether embedded platforms could keep up with the high-end platforms. This is because in a real-world application it is much more desirable to be able to perform VPR on an embedded platform, due to their compact and lightweight design.

To this end a benchmark called Real-time Matched Frames (RMF) was presented to indicate a technique's predicted efficiency and accuracy in a real-time application. This was presented alongside the PR-curves of each technique. PR-curves are widely accepted as the best indicator of a VPR techniques performance, so these were included to identify whether techniques that are traditionally excepted as well performing are better in a real-time application than those traditional accepted as not as good.

CALC produced the highest number of accurate results, indicating it had the most potential for real-time applications. However, several limitations of RMF were noted, as well as some solutions to these limitations in chapter 4.

5.2 Future Work

A prime objective of this thesis is to suggest a number of research ideas and extensions to not only the work presented within this thesis but that will aim to fill other research gaps identified within the field as a whole. To this end, a number of research suggestions and extensions are enlisted below:

1. As mentioned in chapter 4 a better analysis of VPR techniques with a view towards real-time applications could be obtained by combining RMF with the true-positives (loop-closures) distribution proposed by Porav et al. [115]. For applications that require a loop-closure every few meters, as otherwise the localisation error drift becomes too large to handle, the metric proposed in [115] compliments the value proposed by RMF.

2. Another extension to the work presented in Chapter 4 to consider would be a system that selectively applies a high performance VPR technique to frames as they come in, combined with a lower performance VPR system that is computationally lightweight and can be applied to every frame. The system would apply the lightweight techniques to every incoming frame and then determine whether to apply the higher performance technique based on the predicted certainty, or respective similarity score, of the lightweight technique's computed match. Designing a system in this manner would theoretically result in a reduced overall computational complexity in contrast to just applying the high performance technique. In addition, the system would theoretically provide more accurate results in comparison to just using the lightweight technique. In this manner the system would take the best strengths of each technique and combine them.

3. The analysis presented in chapter 3 could be continued as new platforms and platform versions are released. An example of such could be the Raspberry Pi 4, with more onboard RAM to help avoid paging to the micro-SD card.

4. As mentioned in chapter 3, GPU platforms were excluded from the experiment due to a desire to keep power consumption as low as possible. However, with the recent improvements in GPU technology it could be beneficial to repeat the experiment instead focusing on GPU boards.

5. As an extension to both the work of Chapter 3 and to the previous suggestion, it would be beneficial to observe the accuracy vs power consumption trade off of several state-of-the-art CPU focused boards in comparison to state-of-the-art GPU focused boards.

6. The evaluation performed in Chapter 3 was limited to small-scale datasets due to the storage capacity of the embedded platforms. The datasets used, therefore, were not necessarily ideal for including a wide variety of appearance and viewpoint variation in the experiment for the techniques tested to tackle.

This experiment could be further extended to utilize large-scale datasets with a greater degree of appearance and viewpoint variations, with a selection of embedded platforms that possess greater storage capacity or as embedded platforms improve over time.

Bibliography

[1] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford, "Visual place recognition: A survey," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 1–19, 2015.

[2] M. Zaffar, S. Ehsan, M. Milford, D. Flynn, and K. McDonald-Maier, "Vpr-bench: An open-source visual place recognition evaluation framework with quantifiable viewpoint and appearance change," *arXiv preprint arXiv:2005.08135*, 2020.

[3] T. Naseer, L. Spinello, W. Burgard, and C. Stachniss, "Robust visual robot localization across seasons using network flows," in *Twenty-eighth AAAI conference on artificial intelligence*, 2014.

[4] C. Valgren and A. J. Lilienthal, "Sift, surf & seasons: Appearance-based long-term localization in outdoor environments," *Robotics and Autonomous Systems*, vol. 58, no. 2, pp. 149–156, 2010.

[5] A. Pronobis, B. Caputo, P. Jensfelt, and H. I. Christensen, "A discriminative approach to robust visual place recognition," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2006, pp. 3829–3836.

[6] S. Garg, N. Suenderhauf, and M. Milford, "Lost? appearance-invariant place recognition for opposite viewpoints using visual semantics," *arXiv preprint arXiv:1804.05526*, 2018.

[7] A. Ranganathan, S. Matsumoto, and D. Ilstrup, "Towards illumination invariance for visual localization," in *2013 IEEE International Conference on Robotics and Automation*, IEEE, 2013, pp. 3791–3798.

[8] M. Milford, C. Shen, S. Lowry, N. Suenderhauf, S. Shirazi, G. Lin, F. Liu, E. Pepperell, C. Lerma, B. Upcroft, *et al.*, "Sequence searching with deep-learnt depth for condition-and viewpoint-invariant route-based place recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, pp. 18–25.

[9] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, "Simultaneous localization, mapping and moving object tracking," *The International Journal of Robotics Research*, vol. 26, no. 9, pp. 889–916, 2007.

[10] T. Naseer, G. L. Oliveira, T. Brox, and W. Burgard, "Semantics-aware visual localization under challenging perceptual conditions," in *2017 IEEE ICRA*, 2017, pp. 2614–2620.

[11] H. Tian, *Noise analysis in CMOS image sensors*. stanFord university, 2000.

[12] M. Zaffar, A. Khaliq, S. Ehsan, M. Milford, K. Alexis, and K. McDonald-Maier, "Are state-of-the-art visual place recognition techniques any good for aerial robotics?" *arXiv preprint arXiv:1904.07967 ICRA 2019 Workshop on Aerial Robotics*, 2019.

[13] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *ECCV*, Springer, 2006, pp. 404–417.

[14] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV, Springer*, vol. 60, no. 2, pp. 91–110, 2004.

[15] F. Maffra, L. Teixeira, Z. Chen, and M. Chli, "Loop-closure detection in urban scenes for autonomous robot navigation," in *2017 International Conference on 3D Vision (3DV)*, IEEE, 2017, pp. 356–364.

[16] F. Maffra, Z. Chen, and M. Chli, "Tolerant place recognition combining 2d and 3d information for uav navigation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 2542–2549.

[17] F. Maffra, L. Teixeira, Z. Chen, and M. Chli, "Real-time wide-baseline place recognition using depth completion," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1525–1532, 2019.

[18] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.

[19] M. Kalantari, F. Jung, N. Paparoditis, and J.-P. Guédon, "Robust and automatic vanishing points detection with their uncertainties from a single uncalibrated image, by planes extraction on the unit sphere," in *ISPRS2008*, 2008, pp. 203–208.

[20] G. Baatz, K. Köser, D. Chen, R. Grzeszczuk, and M. Pollefeys, "Handling urban location recognition as a 2d homothetic problem," in *European Conference on Computer Vision*, Springer, 2010, pp. 266–279.

[21] L. Teixeira, F. Maffra, M. Moos, and M. Chli, "Vi-rpe: Visual-inertial relative pose estimation for aerial vehicles," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2770–2777, 2018.

[22] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *null*, IEEE, 2003, p. 1470.

[23] R. Arandjelovic and A. Zisserman, "All about vlad," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2013, pp. 1578–1585.

[24] B. K. Horn, "Closed-form solution of absolute orientation using unit quaternions," *Josa a*, vol. 4, no. 4, pp. 629–642, 1987.

[25] L. Kneip, D. Scaramuzza, and R. Siegwart, "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation," in *CVPR 2011*, IEEE, 2011, pp. 2969–2976.

[26] J. Qiu, Z. Cui, Y. Zhang, X. Zhang, S. Liu, B. Zeng, and M. Pollefeys, "Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3313–3322.

[27] W. Van Gansbeke, D. Neven, B. De Brabandere, and L. Van Gool, "Sparse and noisy lidar completion with rgb guidance and uncertainty," in *2019 16th international conference on machine vision applications (MVA)*, IEEE, 2019, pp. 1–6.

[28] J. Ku, A. Harakeh, and S. L. Waslander, "In defense of classical image processing: Fast depth completion on the cpu," in *2018 15th Conference on Computer and Robot Vision (CRV)*, IEEE, 2018, pp. 16–22.

[29] M. Zaffar, A. Khaliq, S. Ehsan, M. Milford, and K. McDonald-Maier, "Levelling the playing field: A comprehensive comparison of visual place recognition approaches under changing conditions," *arXiv preprint arXiv:1903.09107, IEEE ICRA Workshop on Database Generation and Benchmarking*, 2019.

[30] D. Hulens, T. Goedemé, and J. Verbeke, "How to choose the best embedded processing platform for on-board uav image processing?" *Proceedings VISAPP 2015*, pp. 1–10, 2015.

[31] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE T-RO*, vol. 32, no. 6, pp. 1309–1332, 2016.

[32] M. Zaffar, S. Ehsan, R. Stolkin, and K. M. Maier, "Sensors, slam and long-term autonomy: A review," in *2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, IEEE, 2018, pp. 285–290.

[33] W. Burgard, M. Hebert, and M. Bennewitz, "World modeling," in *Springer handbook of robotics*, Springer, 2016, pp. 1135–1152.

[34] J. Klippenstein and H. Zhang, "Quantitative evaluation of feature extractors for visual slam," in *Fourth Canadian Conference on Computer and Robot Vision (CRV'07)*, IEEE, 2007, pp. 157–164.

[35] M. Montemerlo and S. Thrun, "Simultaneous localization and mapping with unknown data association using fastslam," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, IEEE, vol. 2, 2003, pp. 1985–1991.

[36] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós, "A comparison of loop closing techniques in monocular slam," *Robotics and Autonomous Systems*, vol. 57, no. 12, pp. 1188–1197, 2009.

[37] A. Elfes, "Sonar-based real-world mapping and navigation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 3, pp. 249–265, 1987.

[38] C. Evers, A. H. Moore, and P. A. Naylor, "Acoustic simultaneous localization and mapping (a-slam) of a moving microphone array and its surrounding speakers," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2016, pp. 6–10.

[39] M. Kreković, I. Dokmanić, and M. Vetterli, "Echoslam: Simultaneous localization and mapping with acoustic echoes," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Ieee, 2016, pp. 11–15.

[40] J. Djugash, S. Singh, G. Kantor, and W. Zhang, "Range-only slam for robots operating cooperatively with sensor networks," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, IEEE, 2006, pp. 2078–2084.

[41] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Proceedings of the 2005 IEEE international conference on robotics and automation*, IEEE, 2005, pp. 2432–2437.

[42] G. D. Tipaldi, M. Braun, and K. O. Arras, "Flirt: Interest regions for 2d range data with applications to robot navigation," in *Experimental Robotics*, Springer, 2014, pp. 695–710.

[43] S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *2011 IEEE international symposium on safety, security, and rescue robotics*, IEEE, 2011, pp. 155–160.

[44] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *2016 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2016, pp. 1271–1278.

[45] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[46] K. Wang, Y. Liu, and L. Li, "A new algorithm for robot localization using monocular vision and inertia/odometry sensors," in *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, IEEE, 2012, pp. 735–740.

[47] J. Civera, D. Gálvez-López, L. Riazuelo, J. D. Tardós, and J. M. M. Montiel, "Towards semantic slam using a monocular camera," in *2011 IEEE/RSJ international conference on intelligent robots and systems*, IEEE, 2011, pp. 1277–1284.

[48] T. Lemaire, C. Berger, I.-K. Jung, and S. Lacroix, "Vision-based slam: Stereo and monocular approaches," *International Journal of Computer Vision*, vol. 74, no. 3, pp. 343–364, 2007.

[49] P. Elinas, R. Sim, and J. J. Little, "/spl sigma/slam: Stereo vision slam using the rao-blackwellised particle filter and a novel mixture proposal distribution," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, IEEE, 2006, pp. 1564–1570.

[50] L. M. Paz, P. Piniés, J. D. Tardós, and J. Neira, "Large-scale 6-dof slam with stereo-in-hand," *IEEE transactions on robotics*, vol. 24, no. 5, pp. 946–957, 2008.

[51] M. Tomono, "Robust 3d slam with a stereo camera based on an edge-point icp algorithm," in *2009 IEEE international conference on robotics and automation*, IEEE, 2009, pp. 4306–4311.

[52] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald, "Real-time large-scale dense rgb-d slam with volumetric fusion," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 598–626, 2015.

[53] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an rgb-d camera," in *Robotics research*, Springer, 2017, pp. 235–252.

[54] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the rgb-d slam system," in *2012 IEEE international conference on robotics and automation*, IEEE, 2012, pp. 1691–1696.

[55] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for rgb-d cameras," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2013, pp. 2100–2106.

[56] ——, "Robust odometry estimation for rgb-d cameras," in *2013 IEEE international conference on robotics and automation*, IEEE, 2013, pp. 3748–3754.

[57] J.-H. Kim and M. J. Chung, "Slam with omni-directional stereo vision sensor," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, IEEE, vol. 1, 2003, pp. 442–447.

[58] H. Tamimi, H. Andreasson, A. Treptow, T. Duckett, and A. Zell, "Localization of mobile robots with omnidirectional vision using particle filter and iterative sift," *Robotics and Autonomous Systems*, vol. 54, no. 9, pp. 758–765, 2006.

[59] L. Paya, A. Gil, and O. Reinoso, "A state-of-the-art review on mapping and localization of mobile robots using omnidirectional vision sensors," *Journal of Sensors*, vol. 2017, 2017.

[60] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Ultimate slam? combining events, images, and imu for robust visual slam in hdr and high-speed scenarios," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 994–1001, 2018.

[61] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, "Visual simultaneous localization and mapping: A survey," *Artificial intelligence review*, vol. 43, no. 1, pp. 55–81, 2015.

[62] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual slam algorithms: A survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, no. 1, pp. 1–11, 2017.

[63] B. Ferrarini, M. Waheed, S. Waheed, S. Ehsan, M. Milford, and K. D. McDonald-Maier, "Visual place recognition for aerial robotics: Exploring accuracy-computation trade-off for local image descriptors," in *2019 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, IEEE, 2019, pp. 103–108.

[64] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," in *CVPR*, 2016, pp. 5297–5307.

[65] G. Tolias, R. Sicre, and H. Jégou, "Particular object retrieval with integral max-pooling of cnn activations," *arXiv:1511.05879, ICLR*, 2016.

[66] E. Stumm, C. Mei, and S. Lacroix, "Probabilistic place recognition with covisibility maps," in *IROS*, IEEE, 2013, pp. 4158–4163.

[67] A. C. Murillo, J. J. Guerrero, and C. Sagues, "Surf features for efficient robot localization with omnidirectional images," in *Proceedings of IEEE ICRA*, 2007, pp. 3901–3907.

[68] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, IEEE, vol. 1, 2005, pp. 886–893.

[69] W. T. Freeman and M. Roth, "Orientation histograms for hand gesture recognition," in *International workshop on automatic face and gesture recognition*, vol. 12, 1995, pp. 296–301.

[70] C. McManus, B. Upcroft, and P. Newmann, "Scene signatures: Localised and point-less features for localisation," *Robotics, Science and Systems Conference*, 2014.

[71] M. Zaffar, S. Ehsan, M. Milford, and K. McDonald-Maier, "Cohog: A light-weight, compute-efficient, and training-free visual place recognition technique for changing environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1835–1842, 2020.

[72] M. J. Milford and G. F. Wyeth, "Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights," in *International Conference on Robotics and Automation*, IEEE, 2012, pp. 1643–1649.

[73] L. Liu, C. Shen, and A. van den Hengel, "Cross-convolutional-layer pooling for image recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 11, pp. 2305–2313, 2016.

[74] Z. Chen, O. Lam, A. Jacobson, and M. Milford, "Convolutional neural network-based place recognition," *preprint arXiv:1411.1509*, 2014.

[75] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

[76] Z. Chen *et al.*, "Deep learning features at scale for visual place recognition," in *ICRA*, IEEE, 2017, pp. 3223–3230.

[77] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[78] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[79] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[80] A. Babenko and V. Lempitsky, "Aggregating deep convolutional features for image retrieval," *arXiv preprint arXiv:1510.07493 ICCV*, 2015.

[81] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," *arXiv preprint arXiv:1506.02025*, 2015.

[82] Z. Chen, L. Liu, I. Sa, Z. Ge, and M. Chli, "Learning context flexible attention model for long-term visual place recognition," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4015–4022, 2018.

[83] J. M. Facil, D. Olid, L. Montesano, and J. Civera, "Condition-invariant multi-view place recognition," *arXiv preprint arXiv:1902.09516*, 2019.

[84] S. Hausler, A. Jacobson, and M. Milford, "Multi-process fusion: Visual place recognition using multiple image processing methods," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1924–1931, 2019.

[85] A. Khaliq, S. Ehsan, M. Milford, and K. McDonald-Maier, "A holistic visual place recognition approach using lightweight cnns for severe viewpoint and appearance changes," *arXiv:1811.03032, IEEE T-RO*, 2018.

[86] A. Fraser and D. Marcu, "Measuring word alignment quality for statistical machine translation," *Computational Linguistics*, vol. 33, no. 3, pp. 293–303, 2007.

[87] D. Powers, "Evaluation: From precision, recall and f-factor to roc, informedness, markedness & correlation," *Mach. Learn. Technol.*, vol. 2, Jan. 2008.

[88] P. A. Flach, "The geometry of roc space: Understanding machine learning metrics through roc isometrics," in *Proceedings of the 20th international conference on machine learning (ICML-03)*, 2003, pp. 194–201.

[89] J. Fürnkranz and P. A. Flach, "Roc 'n'rule learning—towards a better understanding of covering algorithms," *Machine learning*, vol. 58, no. 1, pp. 39–77, 2005.

[90] D. Powers, "Recall and precision versus the bookmaker," in *Cognitive Science - COGSCI*, Jan. 2003, pp. 529–534. DOI: `10.13140/RG.2.1.3754.1926`.

[91] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *arXiv preprint arXiv:1605.07678*, 2016.

[92] *Nvidia jeston tx1*, `https://developer.nvidia.com/embedded/jetson-tx1`.

[93] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7310–7311.

[94] A. Karki, C. P. Keshava, S. M. Shivakumar, J. Skow, G. M. Hegde, and H. Jeon, "Tango: A deep neural network benchmark suite for various accelerators," in *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, IEEE, 2019, pp. 137–138.

[95] I. Palit, Q. Lou, R. Perricone, M. Niemier, and X. S. Hu, "A uniform modeling methodology for benchmarking dnn accelerators," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, IEEE, 2019, pp. 1–7.

[96] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," *ACM SIGARCH computer architecture news*, vol. 35, no. 2, pp. 13–23, 2007.

[97] R. Sahdev and J. K. Tsotsos, "Indoor place recognition system for localization of mobile robots," in *2016 13th Conference on computer and robot vision (CRV)*, IEEE, 2016, pp. 53–60.

[98] A. Torii *et al.*, "24/7 place recognition by view synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1808–1817.

[99] N. Merrill and G. Huang, "Lightweight unsupervised deep loop closure," *arXiv preprint arXiv:1805.07703, Robotics Science and Systems Conference*, 2018.

[100] M. Milford, "Vision-based place recognition: How low can you go?" *The International Journal of Robotics Research*, vol. 32, no. 7, pp. 766–789, 2013.

[101] M. Larsson, E. Stenborg, L. Hammarstrand, M. Pollefeys, T. Sattler, and F. Kahl, "A cross-season correspondence dataset for robust semantic segmentation," in *CVPR*, 2019, pp. 9532–9542.

[102] M. Zaffar, S. Ehsan, M. Milford, and K. McDonald-Maier, "Memorable maps: A framework for re-defining places in visual place recognition," *https://arxiv.org/abs/1811.03529*, 2018.

[103] A. Abdelbaki, M. Bennewitz, and R. Sabverzavi, "Convnet features for lifelong place recognition and pose estimation in visual slam," Ph.D. dissertation, Mar. 2018. DOI: `10.13140/RG.2.2.10816.89607`.

[104] J. Mount and M. Milford, "2d visual place recognition for domestic service robots at night," in *2016 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2016, pp. 4822–4829.

[105] S. Skrede, *Nordland dataset*, 2013.

[106] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3234–3243.

[107] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[108] H. Badino, D. Huber, and T. Kanade, "Visual topometric localization," in *2011 IEEE Intelligent vehicles symposium (IV)*, IEEE, 2011, pp. 794–799.

[109] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The oxford robotcar dataset," *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 3–15, 2017.

[110] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1452–1464, 2017.

[111] *Up-cht01*, `https://www.aaeon.com/en/p/up-board-computer-board-for-professional-makers`.

[112] *Raspberry pi 3 model b*, `https://www.raspberrypi.org/products/raspberry-pi-3-model-b/`.

[113] *Odroid xu4*, `https://www.odroid.co.uk/odroid-xu4`.

[114] B. Siepert and I. Wellish, *Adafruit ina260 current voltage power sensor breakout*, May 2019.

[115] H. Porav, W. Maddern, and P. Newman, "Adversarial training for adverse conditions: Robust metric localisation using appearance transfer," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 1011–1018.