THE UNIVERSITY *of* EDINBURGH

Analysis of Visual SLAM for Forest Environments

James Garforth

Doctor of Philosophy

Institute of Perception, Action and Behaviour

School of Informatics

University of Edinburgh

2024

Abstract

The objective of this dissertation is to investigate visual navigation for robotics in the forest domain. Visual Simultaneous Localisation and Mapping (SLAM) is a core technology for deployment of robotic systems in many use cases, allowing a robot to build a map of an unknown environment while also keeping track of its own location. Visual SLAM is particularly valuable in settings where access to external positioning data, such as the Global Positioning System (GPS), is unreliable or entirely unavailable. This is true of indoor environments, such as offices, hence most developments of SLAM systems are evaluated against these. However, other GPS denied natural environments such as forests, where canopy cover can block satellite signals, have been left relatively unexplored.

We seek to determine the factors that set the forest environment apart from those more typically explored in SLAM research, establish the applicability of existing systems to these challenging conditions, and investigate processing techniques which might aid adaptation of visual SLAM to forest scenes. We take an approach that aims to identify and qualify challenging factors absent from classic SLAM datasets, and how these factors influence performance, rather than simply presenting a system well-tuned to the task but no more robust.

The first contribution in this thesis is a study comparing performance of state of the art monocular visual SLAM systems on new and existing forest datasets, which confirms the challenging nature of the task. We also propose a suite of visual scene statistics which aim to measure key traits in video data known to cause difficulties in navigation. We show that real forest data presents significantly more of these challenging traits, while simulated forest data fails to reflect the same traits as the real forest. In the second contribution, we take performance analysis further by investigating the impact of initialisation motions in the data and the tuning of system parameters.

Our third contribution presents an improved forest simulation, with a variety of simulated environmental conditions such as wind and snow. Using this simulation, we

are then able to further demonstrate the links between our visual scene statistics and the traits of the environment we proposed they would reflect, as well as independently demonstrate how each trait impacts the performance of a SLAM system.

In the process of this thesis, we also developed a modular computer vision system allowing for rapid configuration of vision pipeline tests, both for evaluating statistics of datasets and for modifying input to SLAM systems. The fourth contribution takes advantage of this pipeline to test the effects of a variety of preprocessing steps on feature matching in forest scenes and overall SLAM performance, opening up potential avenues for improving efficiency.

The final contribution is to assess the performance of place recognition algorithms for SLAM loop closure in forest data. We compare state of the art convolutional neural architecture NetVLAD with popular algorithmic approaches (FABMAP, DBoW2 and VLAD) and find that it generalises better to these new scenes. We integrate this solution with a SLAM system, finding that it proposes correct loop closure locations, but resolving feature matches between these scenes remains a challenge.

Lay Summary

This dissertation focuses on improving the ability of robots to navigate in forest environments using visual information. To achieve this, the study explores a technology called Visual Simultaneous Localisation and Mapping (SLAM), which enables robots to create maps of unknown environments while keeping track of their own location. Visual SLAM is particularly valuable in places where global positioning system (GPS) signals are unreliable or unavailable, such as forests where the dense canopy can block satellite signals.

The research aims to understand the unique challenges that forests pose for SLAM and investigates methods to adapt existing SLAM systems to work effectively in these complex environments. To better understand the factors contributing to these challenges, the research proposes a set of visual scene statistics that measure key characteristics causing navigation difficulties in forests. Real forest data is found to exhibit significantly more of these challenging traits compared to simulated forest data, data from offices and roads.

Further analysis investigates the impact of how the camera is initially moved and system settings on SLAM performance. To support these investigations, an improved forest simulation is created, including various environmental conditions like wind and snow. The simulation helps demonstrate the connection between visual scene statistics and the environment, as well as how each condition affects SLAM system performance.

Additionally, a modular computer vision system is developed that allows for quick testing and configuration of vision pipelines. This system is used to test the effects of different preprocessing steps on feature matching in forest scenes and overall SLAM performance, suggesting potential ways to enhance efficiency.

Lastly, the dissertation evaluates the performance of algorithms for recognising place in forest data. A convolutional neural architecture called NetVLAD is found to work better than other traditional algorithms for these new scenes. While the inte-

grated solution proposes correct place matches, this is not enough to produce a fully functioning SLAM system.

Overall, this research sheds light on the complexities of robotic navigation in forest environments and proposes techniques and insights to enhance the reliability and adaptability of visual SLAM systems in such challenging conditions. By understanding the specific challenges faced in forests, this work brings us closer to deploying robotic systems effectively in GPS-denied natural environments.

Acknowledgements

First and foremost I thank Barbara Webb, for peerless supervision and support. You encouraged me when I cared too little and calmed me when I cared too much, and I would not have completed this chapter of my life without your guidance.

I thank my family, for always believing in me and assuming that if they can't understand what I'm doing it must be clever rather than poorly explained.

I thank my friends, whose praises I could sing forever:

Alex, Amy (and Otto), Annabel, Hazel, Jenny, Rich; old friends who have stood by me through thick and thin.

Alyssa, Benedetta, Charlie, Julie-Anne, Liz x 3, Sigrid, SJ, Susanna, who have moaned about PhDs with me, in between being inspiring people.

Signe, who always brought me gifts at just the right time.

Carson, Raluca, Tatiana, who I was lucky enough to share an office, nachos and endless coffees with. David, Meredith and Elizabeth, who have taken over this role.

Eric and Paola, who were the perfect flatmates.

I thank the Insect Robotics Group (and visitors) whose interesting work provided thought provoking variety to my weeks. Especially to Mike, Tom, Jan, Anna, Andrea and Persie.

I thank my CDT cohort:

- Marco, who has the courage to address something that isn't working.
- Iris, who has the tenacity to fight adversity over and over.
- Jose, who has the resourcefulness to get what he deserves.
- Raluca, who is so good she's getting mentioned twice.
- Daniel, who demonstrated that looking after yourself and achieving don't have to be mutually exclusive.
- Hans, who I can only ever remember laughing.
- Manny, who has the boundless enthusiasm to challenge my cynicism.
- Wolf, who is unstoppable.

- Thibault, who made juggling a PhD and caring for a family seem effortless.
- Teun, who somehow managed to stay happy and healthy the whole way through. I suspect witchcraft.

I thank Bob Fisher and Margarita Chli, who provided supervision but also shared their enthusiasm and wisdom, as well as the folks at Carbomap, particularly Antoine, for inspiring the direction of my work.

I thank Horse Force Four, the bees, and Nikki, for reminding me to thank the bees.

And finally I thank you, the person I inevitably forgot to thank.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*James Garforth*)

To my father, who never doubted that I would do this, even if he didn't live to see it.

Table of Contents

Chapter 1

Introduction

Simultaneous Localisation and Mapping (SLAM) allows a robot to navigate through an unknown environment while building a model of it, and is critical to the success of many robotic applications. A wide range of techniques and full systems have been developed, although successful demonstrations are often confined to static, structured environments with consistent lighting, which simplifies the navigation problem. It remains an ongoing challenge to expand the capabilities of SLAM systems to be robust against motion in the scene, changing environmental conditions, or novel and unstructured environments. So far successes in these areas are still heavily dependent on additional information, sensors, or other careful control of the task such as known routes or placement of easily identified landmarks.

This thesis investigates forests as a domain for visual monocular SLAM. We seek to determine the factors that set this environment apart from those more typically explored in SLAM research, establish the applicability of existing systems to these challenging conditions, and identify processing techniques or other approaches which might aid adaptation.

1.1 Motivation and Requirements

This work was in part motivated through conversation with Geoscientists studying forest health at a large scale. Most survey systems available for gathering data about forests work above canopy level, either from an aircraft or from a satellite, leaving below-canopy information harder to gather (Aguiar et al., 2020). Traditional approaches to gathering trunk widths, for example, rely on measurements taken by hand in only a sample portion of a forest. Our colleagues expressed the value that would come to forest conservation from being able to automate this task cheaply and on a larger scale. To achieve this automation, robotic platforms would have to be able to navigate reliably through natural and managed forests. SLAM systems that have been developed and tested in indoor or highly structured environments are likely to be poorly adapted for redeployment into the wild. While others have deployed SLAM systems in forests before, with limited success (McGlade et al., 2022), to our knowledge no work has investigated the causes of differing performance between domains. Assessing and addressing the adaptations necessary for forest SLAM is the focus of our work.

The choice of forest inspection as the application domain feeds into and focuses the technical direction. For a solution to be deployable at scale and at low cost, we favour compatibility with off the shelf, mass produced platforms rather than bespoke. To avoid the challenges of traversing rough terrain and low-lying vegetation, we focus on flying platforms, particularly micro aerial vehicles (MAVs), as also recommended by McGlade et al. (2022). Finally, to keep the cost of that platform low, this thesis focuses on purely single camera (monocular) visual SLAM and on systems which could run in real time without difficulty when processing capacity is limited.

Rigorous evaluation in our chosen domain proves a particular challenge. Traditionally, to properly compare the localisation or reconstruction performance of a SLAM algorithm requires ground truth data of its track or the 3D structure of the world (Zhang and Scaramuzza, 2018). Neither of these ground truths were available in any existing forest data at the start of our project (indeed, forest datasets are generally rare), and

it is also not trivial to create a dataset that includes them, especially considering the very mapping challenges this work sets out to investigate. As a result, this thesis uses a variety of other techniques for interrogating the performance of the systems: from measuring performance on sub-tasks, like feature tracking, to photorealistic forest simulation. All of these approaches have their limitations, and these are discussed openly throughout this work. We aim to understand what might currently be possible within these constraints, and what characteristics of the forest domain pose challenges to existing SLAM solutions, with an approach that might support forays into other challenging environments in future.

1.2 Research Aims and Objectives

This thesis aims to:

Evaluate the performance of existing state of the art monocular SLAM systems in forest conditions.

Determine factors setting forest environments apart from others more typically explored in SLAM research.

Develop simulation tools for controlling, and evaluating performance impacts of, potentially challenging visual traits of scenes.

Investigate processing techniques which could aid adaptation of SLAM systems for forest environments.

1.3 Contributions

The main contributions of this thesis, by chapter, are as follows:

Chapter 2 reviews background literature setting the context for this work. We introduce concepts in navigation, then the history and state of the art in visual

SLAM research, highlighting particularly notable full systems. We then cover previous work in automating the mapping process for forestry and other forest-based contexts, as well attempts in robotics to adapt to natural scenes.

Chapter 3 presents new forest datasets gathered during this work for testing localisation performance. These datasets present a range of challenge, from fully off-road erratic movement to controlled conditions to aid in SLAM initialisation. We supplement these with a photorealistic simulated forest environment allowing replay of identical routes in varying conditions. This chapter also presents the EnVisionEd modular visual processing system, which allows for rapid prototyping and evaluation of visual pipelines and supported most of the other experiments presented in this thesis.

Chapter 4 presents a conference paper titled "Visual Appearance Analysis of Forest Scenes for Monocular SLAM", published at the International Conference on Robotics and Automation 2019. This paper reviews the performance of state of the art visual SLAM systems on forest datasets of varying difficulty, including our new data. It finds that all systems struggle with this forest environment, with some failing entirely. This work also proposes a set of visual scene statistics with which to measure potentially challenging traits of different environments and uses these to compare forest datasets against more traditional environments for SLAM such as offices and roads. Finally, an early version of a photorealistic forest simulation is compared against real forest data, finding that even though it appears realistic to the human eye it fails to reflect many of the identified visual traits.

Chapter 5 presents a more follow-up investigation into the performance of monocular SLAM systems in forest environments. We create a new forest dataset with ideal initialisation movements to rule out that forest failures are due to movement rather than environment, and show that even in this context systems still

often fail to initialise or encounter failures later on the route. We also experiment with parameter tuning for each of the systems, demonstrating that initialisation failures are also not specifically caused by incorrect parameters. In addition, we look further into where failures are occurring within the code these systems.

Chapter 6 improves upon the simulated forest presented in Chapter 4 to allow the varying of individual environmental traits through cloud, wind and snow weather conditions. This new data is used to assess how well the previously proposed scene statistics reflect the environmental traits we expect them to. We also present here a comparision of SLAM performance between these environmental conditions.

Chapter 7 investigates using preprocessing techniques inspired by animal visual processing to improve feature matching performance on forest data. These experiments have few significant findings, but do suggest some interesting directions for follow up to increase performance or efficiency under challenging conditions.

Chapter 8 has been published as a journal article titled "Lost in the Woods? Place Recognition for Navigation in Difficult Forest Environments" in a special issue of Frontiers in Robotics. It demonstrates the capacity of the NetVLAD convolutional network architecture for describing forests scenes. NetVLAD is compared with state of the art algorithmic approaches used for loop closure detection (FABMAP, DBoW2 and VLAD), outperforming them on challenging forest datasets even in the presence of changing environmental conditions. This work also presents an integration of NetVLAD as the loop closure detection mechanism for a state of the art visual SLAM system and finds that while it is a good solution for correctly identifying loops, the SLAM system is unable to resolve features between proposed views of the same place.

Chapter 2

Background

This chapter introduces background concepts which form a starting point for the work undertaken in this thesis. The individual contribution chapters 4 to 8 then each contain their own background material which introduces the specifics of the work in those chapters. Here we introduce concepts in navigation, the general structure of monocular SLAM systems, the state of the art in this domain, and what previous work has been done in adapting SLAM to forest conditions. We also pose some questions raised by the literature, which this thesis aims to explore.

2.1 Visual SLAM

Simultaneous Localisation and Mapping (SLAM) research addressed the problem that, for most autonomous systems, the available information about both agent location and environment is inherently noisy, incomplete, or otherwise imperfect. As such, rather than having one of these pieces of information as reliable reference for computing the other, SLAM techniques need to estimate both (performing "localisation" and "mapping") simultaneously. While hardware solutions to the localisation part of the problem do exist, particularly in the form of Global Positioning Systems (GPS), these can have limitations in terms of size and cost, as well as being reliant on an external signal that is unreliable in maybe environments such as indoors or under forest canopy (Aguiar

et al., 2020).

Work into real time visual SLAM began on hand-held cameras with Davison et al. (2007)'s MonoSLAM, which combined feature extraction to track motion in the scene and an Extended Kalman Filter (EKF) to estimate the camera's pose as frame-to-frame estimates accrue over time. Another early milestone in the field was Parallel Tracking and Mapping (PTAM) by Klein and Murray (2007). PTAM presented two important insights which continue to to be exploited in SLAM systems since. First of these is that maintaining an estimate of the camera's pose for all frames is memory and processing intensive, given that successive frames contain a lot of redundant information. PTAM instead stores "keyframes" carefully subselected from the full video input to be representative of the whole. Expensive processing tasks like map-building and loop closure (recognising that two places in the map are the same place in order to correct for errors that lead to them being separate) are then performed only with keyframes. The second insight is that frame-to-frame egomotion tracking and map-based keyframe tasks do not need to be performed at the same rates. As a result, the system can be split into two concurrent loops running at different rates, where the tracking thread can be given processing priority to maintain an up to date motion estimate and the mapping thread can take time to properly optimise the map.

While the many individual SLAM systems vary in their construction, it is often helpful to consider them in terms of their tracking and mapping components. The tracking side is concerned with the real-time motion of the camera between frames and with performing local optimisations such as smoothing of potentially erroneous inputs. The mapping component attempts to construct a single consistent model of the environment from potentially all of the frames ever seen, and to perform global optimisations such as correcting the drift that inevitably occurs over time in tracking, by recognising places when they are revisited. We are primarily interested in visual place recognition for this work, but for model construction, and representation an excellent summary can be found in Rosen et al. (2021).

2.1.1 Tracking

The tracking component of visual SLAM systems is essentially performing visual odometry; that is, it involves incrementally estimating the motion of the camera over a series of image frames. There are two major approaches to doing this: Feature-based and Direct.

Feature-based methods detect visual features in each frame, then attempt to match those features between consecutive frames. The camera pose can then be calculated by minimising reprojection error between pairs of features. Features are often detected at salient, reproducible locations in a scene, such as corners (for which the Harris Corners from Harris et al. (1988) are a popular choice). The features are then stored with a descriptor which aims to make them uniquely identifiable if and when they are seen again. In practice, detector and descriptor are often packaged together. SIFT (Lowe, 2004) and the "sped up" version SURF (Bay et al., 2006) are popular choices for their descriptive power, but many have found them too expensive to compute and match on a large scale for real time applications so alternatives have been proposed such as BRISK (Leutenegger et al., 2011) and ORB (Rublee et al., 2011) which trade some descriptive ability for having fast-matching binary representations.

Direct methods forego the use of features and instead work with pixel values directly. These methods calculate camera poses by minimising photometric error of images directly aligned with one another. Direct methods which use the whole image are referred to as "dense", whereas "semi-dense" methods will concentrate only on pixels in regions of high intensity gradient, as the alignment of flat intensity regions involves a lot of ambiguity. Direct methods perform well in low texture scenes and, because they use more image information than feature based methods, their reconstructions of the world are generally more complete. However, photometric alignment is dependent on a constant lighting assumption that does not hold in many environments, making their deployment challenging (Yang et al., 2018). Direct Sparse Odometry (DSO, Engel et al. (2017)) attempts to address this by using a photometric camera calibration

to recover irradiance images, which has been shown to improve performance indoors. Direct methods are also more sensitive to geometric noise (like the effect of rolling shutter cameras) and so will fair poorly compared to feature based methods where a more expensive global shutter camera isn't available (He et al., 2020).

Another common assumption in SLAM research is rigidity; that the environment stays mostly static and scene motion is predominantly due to the motion of the camera. This has made adapting SLAM systems to environments where this assumption is false challenging. One such case is the work of Scona et al. (2018), where movement of subjects within the scene means they are a dominant source of motion. The authors propose a method for segmenting background and foreground based on clusters of similar motion, so they can be considered separately. Another challenging domain for a rigidity assumption is surgical "intracorporeal" data, where the entire environment is deformable. SD-DefSLAM by Gómez-Rodríguez et al. (2021) is a recent example of dealing with surgical imagery, through a combination of identifying and masking out moving objects with a neural network, and estimating deformation over time with a Shape-from-Template algorithm (Lamarca and Montiel, 2018). Such approaches show promise, but are restricted to very simple environments consisting of essentially one deforming surface.

2.1.2 Place Recognition for Loop Closure

Due to the compound effect of even the smallest pose inaccuracies over the trajectory of a robot, it is inevitable that sometimes the robot will believe it is in a new location when it has actually returned to somewhere previously visited. As a result, features seen in new frames will be attributed to new landmarks, rather than being the correct correspondence with known ones. It is very useful, therefore, for SLAM systems to have some way of detecting when it has made a loop, recognising a place independent of its pose prediction, in order for it to optimise and correct for the drift in its predicted position. Many systems have found this difficult, and it is fairly common in the

literature to see that loop closure is not addressed (Klein and Murray, 2007; Holmes and Murray, 2012). One reason for this is that an incorrect loop closure can severely corrupt a map, so many would rather not take the risk if they can't be confident it will be done robustly.

Williams et al. (2009) provide a useful comparison of the three main types of loop closure technique for monocular SLAM. The types are:

- Map to Map, in which correspondences are searched for between sub-sections of the map.
- Image to Image, where a new frame is compared with past frames for similarity.
- Image to Map, where the new camera image is compared with a projection of the known map into the currently predicted camera frame.

They compare specific implementations of each of these types used for loop closure detection in the same SLAM system, but it should be noted that it would be very difficult to be sure the choice of implementation wasn't partly responsible for differences in results. Williams rules out map-to-map loop closures for monocular SLAM due to the maps being too sparse to contain enough information for comparison (although whether sparse maps are intrinsically required by monocular SLAM is debatable). Between image-to-image and image-to-map, the latter was found to have higher accuracy and, critically, produced no false positives.

Implementations like that of Ta et al. (2013) use a method called "small blurry image" to recognise potential loop closures. Instead of an expensive comparison with every high-resolution keyframe seen before, this method compares extremely low resolution versions of the keyframes with the current frame. Obviously this method is liable to confuse similar looking areas (a big problem in repetitive environments like corridors) but it can be used as a quick first pass, to then be further checked by a more expensive but reliable method.

Williams et al. (2011) utilises a randomised lists classifier. This classifier requires training, which for best results is done offline. This implementation is, however, also able to take advantage of the fact that even a brief viewing of an object with a 30Hz video camera can gather a large number of images and, combining this data with artificial warping, allows for the production of many more samples. In performance trials, with sufficient opportunity for training, the recall is better than SIFT.

Inspired by analogy to text retrieval of the kind done by Google and other search engines, "Bag of Words" systems such as FABMAP (Cummins and Newman, 2008) use a vocabulary of trained "visual words" to describe and compare images. This requires a training set of many images of the kind we are expecting to encounter; this is a potential hindrance to autonomy as we can then only deploy the robot in environments similar to those it has been trained on. Features extracted from the training data are mapped to a descriptor space. A clustering method is used to partition the descriptor space into clusters of similar features, which constitute the visual words. When a new feature is detected, it too can be mapped into the descriptor space and subsequently classified as a given word based on the cluster it is closest to. This way, an image can be summarised as a set (or "bag") of words, one for each of the features extracted from it and two bags of words can be compared to determine the similarity between two images. It is worth noting that the bag of words description will be independent of the spatial relations between those features, so two images with the same features as each other but with their positions jumbled up will still match perfectly.

Unfortunately, even with images summarised as bags of words, when a database of past images for comparison is continually growing it starts to become infeasible to do a comparison with all of them. One solution for this problem is to facilitate the search via a tree built using hierarchical clustering on the descriptor space. The idea of this "vocabulary tree" is that each leaf is equivalent to a word and is associated with each of the images that contains that word. When we have a new feature, we can quickly use its location in descriptor space to navigate down the tree, where we will find all of

the images that also contain that word. Repeat this process for each feature in a new image and we can see which image comes up most frequently: this is the one with the greatest similarity to our new image. Assuming the match is considered good enough (perhaps by the number of features matched between the two images), we can say we have detected a loop closure between the current location and the location in which the matched image was taken.

2.1.2.1 Navigation As Place Recognition

Some works blur the line between place recognition and SLAM, building up a graph of remembered places over time that essentially form a topological map. Localising becomes a case of establishing whether you are in one of the previously seen places and, if not, adding a new place to the map. FABMAP 2.0 (Cummins and Newman, 2011) does this, describing itself as "Appearance-only" SLAM. Basic FABMAP, which performs recursive Bayes estimation over all known places, has been adapted (FABMAP 2.0) to be able to cope with continually growing maps by using an inverted index that allows quick retrieval of images containing a particular visual word, and modifying the probabilistic model to account for not having access to all observations. FABMAP 2.0 is shown to have similar precision-recall results to original FABMAP, but benefits from being orders of magnitude faster, especially on large maps.

Milford et al's SeqSLAM (Milford and Wyeth, 2012) takes a global descriptor rather than feature-based approach to place recognition SLAM. They heavily downsample keyframe images to 64 by 32 pixels, convert to greyscale and perform patch normalisation. Local contrast enhancement is also applied to similarities, and matches are searched for as sequences of local best matches rather than a single global best match. The normalisations performed by SeqSLAM make it good at matching scenes by their structure and how their appearance changes as the camera moves, rather than by similarity between single pairs of frames. This focus on overall structure rather than appearance means that SeqSLAM handles lighting change and place recognition across

seasons very well, as shown in Sünderhauf et al. (2013). The down side, however, is that this method requires the motion taken through the scene to be similar each time it is viewed, so it is primarily useful for applications like cars or trains, where places can be reasonably expected to be traversed along a set path at a roughly constant velocity.

Given SeqSLAM's successes, a number of works have attempted to extend it and deal with its limitations. Johns and Yang (2013) present their Feature Co-occurence Maps, which takes the sequence based matching approach but switch back to the FABMAP approach of using local features and a BoW model instead of global descriptors. In addition to describing a scene with visual words, they also use "spatial words", which describe the geometric relation between pairs of visual words. Looking at the presence and arrangement of local features rather than the structure of the whole image makes this method more robust to changes in viewing angle than SeqSLAM. When searching for sequence matches, Johns et al. consider all forward moving paths rather than fitting a straight line, avoiding the assumption that the camera is not accelerating.

Lynen et al. (2014) "Placeless" place recognition also switches to using local feature descriptors (BRISK), but with a kNN voting system rather than a bag of words for matching. The sequences compared to each other are dynamically defined based on density of features to avoid the bias towards complex scenes that voting systems often suffer from. The algorithm uses statistical tests rather than thresholds to determine matches, meaning that there is only a single parameter to set up for their system. This is currently a batch method, not able to be used real-time. The authors mention producing an online version, but though they have made improvements such as in Lynen et al. (2017), this has not yet been among them.

2.1.2.2 Deep Learning and Place Recognition

Image matching is a popular task for Deep Neural Networks (DNNs), and so it is little surprise that DNNs have been oft-explored to place recognition tasks, demonstrating

robustness to changes in appearance (Gomez-Ojeda et al., 2015), lighting (Gomez-Ojeda et al., 2018), and viewpoint (Chen et al., 2017). In the case of Kendall et al. (2015), Walch et al. (2017) and more recently Li et al. (2019), many systems treat the task as outputting a 6 degree of freedom pose regression respective to the original ”map” of images. What this means is, however, that the domain in which these systems can localise is restricted to the area covered by the original image dataset. This makes them poorly suited to many real world exploratory tasks, where the place being recognised may have only been seen since deployment.

The more useful systems for real world place recognition are those such as NetVLAD (Arandjelovic et al., 2016), which produce a descriptor of a place designed to be efficiently compared with those other places the system has visited, similar to non-DL approaches like FABMAP (Cummins and Newman, 2008). NetVLAD combines a layer replicating the descriptive power of VLAD descriptors (Jégou et al., 2010) on top of features learned from training on datasets of millions of scene images. The result is a network trained to produce feature vectors where the euclidean distance between two compared vectors should be smaller when the images they were produced from observe the same scene. NetVLAD’s descriptors are small enough to be plausibly used for real time place matching, and have been shown to be robust to a lot of visual appearance change.

The CNN-based method of Chen et al. (2017) moves away from whole-image descriptors, instead describing patches within images in a way that makes them robust to partial views of scenes. Hausler et al. (2021) combines this approach with NetVLAD to produce patch VLAD descriptors. Vysotska and Stachniss (2019) have taken what could be considered the opposite approach and attempt to match not single image descriptors but sequences of image descriptors. In this way they adaptively define what a ”place” is in each context, and are much better prepared to work with recognition in continuous video data.

2.1.3 State of the Art Monocular SLAM

ORB-SLAM (Mur-Artal et al., 2015) attempts to build upon the main ideas of PTAM, while addressing its limitations. Their system is built around the use of ORB features (Rublee et al., 2011), which are extracted once but used for the tracking, mapping and loop closure subsystems, in order to reduce necessary processing. ORB features provide the system with very fast feature comparison, as well as greater invariance to viewing angle than PTAM. While PTAM requires a specific motion with respect to a planar scene when it starts in order to initialise a baseline, ORB-SLAM does this automatically by detecting the type of scene being viewed and using a suitable initialisation method. It stays in this initialisation procedure until it achieves significant parallax. Once running, ORB-SLAM upholds a survival of the fittest approach to keyframes. The requirements to add a new keyframe are very low and don't require a minimum distance to have been covered between frames, only a minimum amount of visual change. The result is that, unlike PTAM, ORB-SLAM will continue to add keyframes when motion is primarily rotational. In order to compensate for the influx of additional map points for which triangulating depth may not be possible, ORB-SLAM aggressively culls map points that are not repeatedly seen soon after creation. Likewise, the additional keyframes are culled by removing those which observe, almost exclusively, map points observable in other keyframes.

Loop closure is given its own thread in ORB-SLAM, on top of the two threads used by PTAM. Image frames are represented as a bag of words vector and compared against a reduced overlapping subset of keyframes called the covisibility graph. When a loop closure is detected, a different, minimal skeleton graph called the "essential graph" is updated to distribute the detected error, with the map points then updated according to one of the essential graph keyframes from which they are observed.

Mur-Artal et al. (2015) observe that ORBSLAM displays tracking accuracy even higher than dense methods like LSD-SLAM (Engel et al., 2014), in spite of the large amount of additional information such methods use. One possibility for this is that due

to the increased computational demands of dense methods, information is not utilised as fully as it could be, for example because map optimisation is done incrementally. Another is that due to the surface reflectance model assumed by dense systems, pixel matching must be done over a narrow baseline, which conflicts with the need to have a wide baseline to accurately estimate depth. An extension of ORB-SLAM is demonstrated in Mur-Artal and Tardós (2015) which builds a semi-dense map on top of the original feature-based one, in order to get the best of both worlds. ORBSLAM has seen wide use and adaptation, and the authors themselves have produced two follow on versions. ORBSLAM2 expands the supported sensor setups to include stereo and depth cameras (Mur-Artal and Tardós, 2017), while ORBSLAM3 integrates inertial sensors and expands to being able to maintain multiple maps, which can be fused as necessary when matches are found between them (Campos et al., 2021).

Leutenegger et al. (2015)'s "OKVIS" uses batch nonlinear optimisation rather than recursive as the method for state estimation from visual and inertial measurements. This batch approach limits linearisation errors and has shown greater precision, but has traditionally been too computationally demanding to run in real-time. Combining camera with inertial data, as in OKVIS and VINS-Mono (Qin et al., 2018), helps to improve local motion estimation, but requires carefully co-calibrated sensor data. This data is absent from most existing datasets, limiting a thorough analysis of visual-inertial SLAM performance.

Semi-Direct Visual Odometry (SVO) (Forster et al., 2014) is a hybrid SLAM system, and combines feature extraction with FAST features with employs a direct method to track features and any pixel with nonzero intensity gradient between frames. The camera trajectory and 3D structure are optimized using reprojection error. SVO stands out for its exceptional efficiency, but, as a pure VO method, it only provides short-term data association, resulting in limitations to its accuracy over longer periods (Campos et al., 2021).

Direct Sparse Odometry (DSO) (Engel et al., 2017) is a camera pose estimation

system known for its ability to perform well in scenarios where traditional point detectors may face difficulties, leading to enhanced robustness in low-textured areas and against blurred images. DSO achieves this by introducing a local photometric bundle adjustment technique that optimizes a window of seven recent keyframes and the inverse depth of sparse 3D points. Noteworthy extensions of DSO include the capability to use stereo or visual-inertial sensor setups, and loop closure using features and DBoW2 (Gálvez-López and Tardós, 2012). These extensions have demonstrated the versatility and adaptability of DSO across various applications within the field of SLAM.

DROID-SLAM (Teed and Deng, 2021) is the first notable instance of a SLAM system built primarily around neural networks, using one for feature extraction and correlation, along with another as an "update operator" to estimate pose and depth updates. The system is trained on monocular data, but shown to be able to handle stereo or RGBD inputs without retraining. It performs very competitively with classic SLAM systems like those above on a range of datasets, although not including much variation in environment type.

2.1.4 Evaluation of SLAM Systems

The main attribute being evaluated when SLAM systems are tested is usually the quality of the trajectory estimate, which is the estimated camera pose values over time for a whole route. This is the same as the procedure for evaluating a Visual or Visual-Inertial Odometry system (as explained by Zhang and Scaramuzza (2018)). To assess a SLAM-generated trajectory, the experimenter requires a ground truth for camera poses obtained from an external source such as GPS or a calibrated camera rig mounted in the experimental environment. The test and ground truth trajectories need to be in the same frame of reference, and thus in many cases it is necessary to calculate a best estimate transform to align them. The demands of trajectory evaluation mean that SLAM testing is often limited to environments that can provide the required data, including

frequent use of simulators such as AirSim (Shah et al., 2017) for roads and ICL NUIM Handa et al., 2014 or Habitat (Szot et al., 2021) for indoor scenes. The ”sim2real” gap, as it is often called, has been identified primarily in machine learning work where simulation would be ideal for running trials quickly and safely generating policies for control (Hfer et al., 2021), or for training visual systems (Doersch and Zisserman, 2019). It is important that these policies then transfer to real world environments for this process to be worthwhile, but many works have identified poor performance and attempted to bridge the gap by changing the learning approach (Doersch and Zisserman, 2019), developing methods to convert policies between the two domains (Sadeghi et al., 2018), or by trying to increase the similarity of the simulation to the real world (Kadian et al., 2020). Despite this work, as noted by Park et al. (2023), there is no established standard for measuring the realism of images produced by a simulator.

In a summary of the field of “Spatial AI”, Davison (2018) outlines a number of other important evaluation metrics for SLAM systems. These include tracking and relocalisation robustness, latency, power usage, data usage or movement, and also the quality of additional information added to the map, such as distances or object labels.

2.2 Navigation In Natural Environments

2.2.1 Mapping for Forestry

In order to perform proper forest management, it is necessary to be able to regularly gather data that can be used to assess a forest’s health. Until recently this task would be performed manually by people physically walking through forested areas and would be both time consuming and partially qualitative, often done only for a sample section of the forest and the findings extrapolated from there.

Automation is starting to become more prevalent in Geospatial analysis, and forestry is benefiting from this trend. Planes and fixed-wing UAVs with downward facing LIDAR sensors are flown over forested areas to quickly gather a great deal of data, such

as in Lefsky et al. (1999), which analyses rich structural and biophysical properties of forests canopies. Such systems are expensive, however, and their ability to accurately investigate the forest below the canopy is limited.

For the measurement of data below the canopy level, the available collection methods are generally still lacking in automation. In order to get tree trunk diameter data, for example, to be used to approximate biomass, the standard approach is still to send a team of people with measuring tapes to a sample region of the forest and assume that region is representative of the average. Where sensors are used for this task, most often it is a scanning sensor head on a stand. Takashi et al. (2014) use such a system with a scanning laser, placed at intervals by the operator. The resulting point clouds have their points clustered into what are assumed to be individual trees, and correspondences between these trees and those already in the map are established. Finally, ICP is applied to fine tune the alignment and trunk diameters at a fixed height are extracted.

Many other systems seeking to map forests also do so by specifically detecting individual trees. Forsman et al. (2016) use high resolution laser scan data and build their map by fitting circular features objects fit to the curve of the visible sections of tree trunks. This methods takes advantage of the general symmetry of tree trunks to create features that can be matched even if a tree is subsequently seen from a completely different direction. Kukko et al. (2017) and Pierzchała et al. (2018) extend this work by using loop closure techniques from SLAM to counteract the problem of drift in the original system.

Miettinen et al. (2007) developed a LIDAR based SLAM system mounted on a mobile platform (an ATV). They find it practically very difficult to match individual trees in all but the sparsest managed forests and instead opt for using trees groups as features. Ohman and Kosti Kannas, Jaakko Jutila, Arto Visala and Pekka Forsman (2008) extend the same work by adding cameras, as the visual appearance of bark on tree trunks makes them easier to differentiate. Most recently, Tang et al. (2015) perform SLAM with a small footprint LIDAR, beating a satellite based system, but

note that this only works in mature forests where there are enough trunk features.

More mobile, online LIDAR systems also often extract only tree trunks from the scene, such as in Schultz et al. (2016) or Liao et al. (2016), as well as for RGBD systems in Fan et al. (2020) and from MAVs in Tian et al. (2020). This avoids much of the complexity of the forest scene but also limits the usefulness of the final map to only information about placement of trees, and limits the usefulness of the system to only forests. While recent works show impressive results for improving laser-based mapping of the whole scene (Tinchev et al., 2018; Liu et al., 2022), laser-based sensors are still expensive and need a lot of power, limiting their use on flying platforms for anything other than very short flights.

2.2.2 Robotics and Natural Scenes

Most robotics research focuses on navigation problems in the environments that humans most frequently use. Office spaces and corridors, along with roads, are the most commonly seen use cases for autonomous mapping robots, as seen in Mur-Artal et al. (2015); Milford and Wyeth (2012); Newcombe et al. (2011) and almost every other paper referenced in Section 2.1. Below we give an overview of works in robotics that have addressed problems related to navigation in natural outdoor environments involving trees.

With the recognition that navigation systems need to handle outdoor environments differently to indoor ones, works such as that of Collier and Ramirez-Serrano (2009) use a visual classification system to determine which kind of environment their robot is in. They then go as far as to switch between two completely separate mapping systems; a laser scanner for indoors and GPS plus IMU for outdoors. Less extreme and more flexible is the approach of Asmar et al. (2004)'s "SmartSLAM", which uses the environment classification (performed by a neural network) to change the types of features it attempts to detect and match, aiming to always be utilising a feature type suitable to the environment, but still building up a single consistent map. While this

work is more of a proof of concept than an extensive study, they suggest that point features are suitable for office environments, while trees might be a better feature to use in a park.

Works such as Ross et al. (2013) have looked at how to reliably perform the obstacle avoidance part of the navigation task in cluttered environments, when limited to a single camera as input. They use imitation learning techniques on demonstration flights performed by human MAV pilots to train a system that avoids trees when flying in a forest, but they do not perform any kind of map building. Other works have looked at the problem of visual trail following under forest canopy, starting with Giusti et al. (2015), then more recently Maciel-Pearson et al. (2018) and Niu et al. (2023). These represent only reactive navigation, rather than mapping, but what they do show us is the potential of neural network based approaches for cutting through the visual complexity of forest scenes. Silva et al. (2020) have recently augmented a trail following system by integrating depth information extracted from LSD-SLAM (Engel et al., 2014), but do not evaluate its performance. Likewise, Smolyanskiy et al. (2017) use DSO for obstacle detection and avoidance in a forest trail setting but perform no evaluation of the resulting track or map.

When it comes to actually building maps of forests with robots, there is little in the literature more advanced than that discussed in the previous section. Aguiar et al. (2020) perform a survey of robot localisation techniques for agriculture and forestry, and report few systems utilising cameras and that even these simplify the problem into 2D, with most relying on GPS integration to assist vision, or not building a map. Even recent precision agriculture mapping works by Chen et al. (2018) or Shu et al. (2020) take advantage of the sensor's movement in a straight line along a row to simplify the problem significantly.

Asmar et al. (2005, 2006) extend upon the environment identification work in SmartSLAM (Asmar et al., 2004) to propose a method for detecting trees in images, which they propose for use in SLAM, but they do not discuss any way of describing

or matching these trees, meaning their system is currently not useful for more than segmentation.

Duggal et al. (2016) work with MAVs as a platform for monitoring crops in plantations. Their system uses a Bebop MAV, fusing ORB-SLAM with GPS and IMU data for navigation. Though this plantation environment is a lot tidier and more ordered than forests, this work suggests that ORB-SLAM might be a good place to start to build a system that can map natural environments reliably.

Chahine and Pradalier (2018) compare three state of the art SLAM systems (ORB-SLAM, DSO and LSDSLAM) on a dataset recorded along the banks of a lake. This environment certainly contains natural scenes, but as the camera always moved along rather than through them the complexity is reduced. Still, the authors describe failure cases to include lens flares (for DSO), initialisation errors (for ORBSLAM) and false recovery after tracking is lost (for LSDSLAM).

At the end of their recent review of the area, McGlade et al. (2022) propose four priorities for research into the use of low cost imaging in forestry. These are 1) methods for integrating sensors into wearable devices, 2) methods for adapting capture procedures to forest settings, 3) SLAM systems for forestry and 4) using low cost devices at terrestrial and airborne levels to make data captures at the scale of whole forest plots. While we do not address 1), the remainder of these points are all reflective of the intentions behind the work presented in this thesis.

2.3 Questions Raised

Our reading of the literature raises some key questions that prompt the work presented in this thesis. These are:

- How well can existing state of the art SLAM systems, such as ORBSLAM and DSO, perform in a forest setting? Can they do this “off the shelf” or do they require tuning?

- What variations in environments' visual appearances (such as the presence of large amounts of foliage) are important variables in performance? Are these differences missing from standard benchmarking datasets?

- Can we produce measures of these variations, ideally running in real time such that they might actively aid SLAM?

- Can we design development or evaluation setups where these environmental traits can be controlled independently?

- Are specific SLAM subsystems, such as feature tracking or loop closure, more or less robust against environmental variety?

Chapter 3

Methodology

3.1 Data Collection

Evaluation of computer vision techniques requires datasets of image or video data replicating the conditions under which the task being investigated might occur. At minimum for navigation tasks like SLAM, this means continuous video from a robot platform as it traverses an environment, while for place recognition it requires discrete sets of images where the same "places" are depicted multiple times. Depending on what traits are needed for a given evaluation, there are many publicly available datasets for both navigation and place recognition.

This section describes the datasets chosen and created for the work reported in this thesis, along with the challenges faced and decisions made during that curation process.

3.1.1 Ground Truth Challenges

Several issues make rigorous performance evaluation difficult for mapping algorithms in a forest environment.

To evaluate estimations of camera pose requires a ground truth of the camera's actual pose, which is usually obtained from more accurate external sensors. Indoors

these could be a collection of calibrated cameras mounted on the walls, and outdoors it will often be a position signal from a GPS system. Placement of external sensors over a large area of forest is costly and difficult to calibrate, in addition to the challenge of hiding those sensors so that they don't become useful features in the collected data. GPS can be collected in forests, but canopy cover interferes with the signal and it very quickly becomes too inaccurate to be useful.

If a SLAM system is also going to be evaluated for its ability to reconstruct the 3D structure of a scene, this also requires a ground truth. In some cases, the environment is simple enough (such as an empty room) to generate a ground truth from measurements separately. More complex or comprehensive 3D ground truths require the use of a more powerful mapping system than the one being evaluated, such as using large powerful LIDAR scanners and offline processing. Mapping forests with even these systems remains an open problem, however (as shown in Risse et al. (2018)), so for now they should not be relied upon for error-free ground truth.

Alternatively to real world data, simulation can provide pose and reconstruction ground truths without error or challenging collection. Additionally, simulated data can potentially vary other difficult factors such as lighting and wind conditions to a level of granularity it is impossible to achieve in the wild. The down side of this approach is that feature rich simulations can be challenging to build. Including challenging real world conditions requires them to first be identified, and simulations run the risk of not properly reflect real data in unpredictable ways. If these differences then impact on evaluation performance, then the simulation is failing to act as a proper benchmark for real world performance of tested algorithms.

3.1.2 New Forest Data

We recorded data from a number of platforms over the course of this work. An outline of each follows, along with a description of the site: Hillwood.

3.1.2.1 Hillwood

All new datasets created for the work contained in this thesis were recorded at the same location: Hillwood Park, a nature reserve on Corstorphine Hill, Edinburgh (55.95486, -3.27363). This wooded hill is easy to access by public transport from the University, so taking equipment there was straightforward. The regions explored for data gathering are all within the area labelled as "W10" by the heritage service managing it (of Edinburgh Council, 2017). This area is around 21 acres of mixed broadleaf woodland, consisting of native ash, oak and elm, with more recent introduction of sycamore, lime, Norway maple, horse chestnut, sweet chestnut, beech and conifers (Wikipedia contributors, 2023). Legal restrictions must be taken into account when gathering MAV data, most importantly a requirement to avoid members of the public. Thankfully, very few other members of the public use the park during week days, so disruptions waiting for people to clear the area are kept to a minimum. Recording is a minimum two person effort, where one person focuses on controlling the MAV, while the other carries the laptop used as the base station and monitors the status of the recording. API is provided by the bebop_autonomy (AutonomyLab, 2018) package, the controller is an XBOX 360 wired controller, and recordings are made through the Robot Operating System (Quigley et al., 2009) into a ROSbag file.

The earlier videos at this location (Hillwood AR) sample from a few places in the forest, including a loop primarily along footpaths. The second return to the site for Hillwood Bebop included recordings of the same looped area again, for comparison, although windy conditions meant it was done in smaller sections. This loop location was chosen emergently as part of the initial exploration, and was not ideal from the point of view of wanting to avoid public footpaths (unnecessary man-made structure) or ease of access (parts of it involved steep slopes, rocks and lying trunks to climb around).

For later loop recordings (which would form Forest Loop) another location within the forest was chosen. This second loop struck the balance of being mainly off-path

Figure 3.1: Satellite image of Hillwood Park, where our data was gathered.

Figure 3.2: Approximate outline of Forest Loop route, with the nearest landmark (Corstorphine Hill Tower) for reference.

(crossing it only once) without having any terrain experimenters would find it hard to traverse. We chose a route that kept us in close proximity to trees, to avoid data that was simply another wide road or corridor with vegetation only at a safe distance, as in existing data. The route followed was mostly rectangular, as much as the layout of trees would allow, to make it easier for us to visually approximate in SLAM reconstructions whether the general motion was being reproduced.

3.1.2.2 Gathering Method

In all data gathering, we are mindful of the motion of the camera to create datasets that reflect expected use case conditions for an autonomous robot in a forest environment, without making the data display traits that make navigation difficult in even controlled lab conditions.

A forward facing camera is necessary for obstacle avoidance, but motion across the visual field rather than expansion is preferred for feature tracking. The platform is yawed at an offset to the direction of travel, but such that the point of expansion is usually still within the image. To further provide useful translation information, linear sections of track are repeatedly crossed left and right as they are traversed. The purely rotational movement of turning on the spot is avoided, in favour of starting turns early

and rotating while continuing translational movement.

There are two main traits we look for in the environment to make sure we are not biasing the kind of data recorded. Firstly, data should be recorded away from the edge of the treeline, such that the forest extends in all directions. Secondly, we minimise the presence of clear tracks or any other man-made structure (e.g. signs, walls) in the videos.

Some initial data was recorded from a smartphone camera to assess the suitability of the environment and trial run SLAM systems before more complicated data gathering with MAVs, but this data was ultimately not included in any of the experiments reported in this thesis. The resolution of the phone camera is higher than any of our MAVs but, as the phone is hand-held, the video does not have the characteristic motion of a flying platform, and particularly noticeable is a "bobbing" motion with footsteps over rough terrain.

3.1.2.3 Parrot AR Drone

The first robotic platform we had available for data collection was Parrot's AR Drone 2.0, a commercial off the shelf four rotor MAV designed for playing augmented reality games via control from a smartphone. The AR Drone's low price point, on board flight stabilisation and well supported API have made it a staple in vision and control research. The AR Drone features a forward facing 1280x720 pixel camera (published at 320x240) with 93 degree field of vision for primary navigation, along with a downward facing camera mostly used by on board flight stabilisation mechanisms. It also has an IMU, GPS and ultrasound / air pressure height sensors.

The software library ardrone_autonomy (AutonomyLab, 2014) is a third party package providing access to all of the elements of the AR Drone API via the Robot Operating System (ROS). We used this package for our data gathering setup, allowing us to pilot the MAV with a hand held controller and to save camera and IMU data to a ROS bag file. This bag can be played back at a later date to recreate the sensor experience

(a) Parrot's AR Drone.

(b) Parrot's Bebop Drone.

of a real flight without the need to run all tests in the field.

3.1.2.4 Parrot Bebop Drone

The Parrot Bebop Drone is a successor to the AR Drone aiming towards the aerial photography market. As a result it has some key domain specific features, in addition to the standard incremental improvements of other components. The ROS package bebop_autonomy (AutonomyLab, 2018) provides the same functionality for this platform as its equivalent does for the AR Drone.

The Bebop has a high resolution (4096x3072) fisheye camera mounted at a downward tilt. Onboard processing subsamples from this wide angle image to produce a video feed that is artificially stabilised but still high resolution (1920x1080), published at 856x480. Part of the initial appeal of this platform was the possibility of utilising the full fisheye image (for a more animal-like field of vision), but unfortunately Parrot does not provide support in the Bebop's API for retrieving this image. We have discussed with other groups who have successfully installed their own firmware on the platform, but they warned that this process is risky so should only be pursued if the wide field of vision is absolutely necessary, which it did not end up being.

3.1.3 Photorealistic Simulation

Intending to have an evaluation environment where we had more fine tuned control over conditions like wind, and where we could run many more iterations of experiments than practical in the real world, we explored the possibility of constructing a

realistic forest simulation. The simulation is based in the Unreal Engine (Epic Games, 2018) and utilises a package of photorealistic forest assets created for film and video game rendering purposes (GmbH, 2022).

The platform used for developing and running the simulated environment is by necessity of higher specification than that used for SLAM experiments. It's key characteristics are as follows:

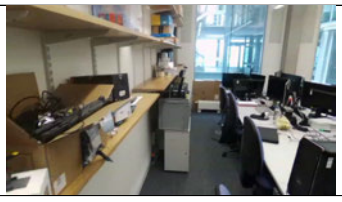
- Processor: AMD Ryzen 5 5600X 6-Core Processor 3.70GHz
- GPU: NVIDIA GeForce GTX 1060 6GB
- RAM: 32.0GB
- OS: Windows 10 Home

For initial data recording, the virtual camera is flown manually through the scene using the mouse and keyboard, and the scene is recorded to video file. Although in principle this should resemble the Hillwood data (a drone travelling through unstructured forest) we note that in practice this produced much smoother motion than a real MAV. In the work described in Chapter 4, we find these recordings to be a poor reflection of real forest video.

3.1.4 Dataset Summary

The work presented in this thesis uses a variety of existing and newly created datasets for assessing navigation performance over a wide range of environments. We've chosen datasets to represent common research environments containing a lot of human structure, such as TUM Monocular (Engel et al., 2016a) and KITTI (Geiger et al., 2012), as well as one of the few examples of a forest dataset in SFU Mountain (Bruce et al., 2015). The full set of datasets is listed below.

Name	Example	Framerate	Resolution	Videos
Hillwood AR		15 fps	320 x 240	3
New forest data from the Parrot AR Drone. Three routes through the same area of Hillwood in the autumn. Canopy is relatively open, and leaf litter is frequently in motion due to wind or MAV down draft. Video 1 follows a half loop before turning and completing the full loop in the other direction. 5 minutes 40 seconds. The other two videos attempt the same loop but wind conditions cause them to crash after approximately 1 minute.				
Hillwood Bebop		30 fps	1920 x 1080	1
New forest data from the Parrot Bebop Drone. Single routes through an area close to Hillwood AR in the autumn of the following year. Canopy is relatively open, wind is calmer, but down draft still moves leaf litter at the start of the video. Flies out from a central point 3 times before returning, no overlap except at the centre. 6 minutes in length.				
Forest Loop		30 fps	856 x 480	2
New forest data recorded from the Parrot Bebop Drone. Different area to the previous recordings, chosen to allow for a clear loop without too much overhanging foliage but also no obvious path. Two versions recorded, identical routes each consisting one complete loop, one of which has the platform carried by hand instead of flying as a comparison and backup. Each approximately 7 minutes and 30 seconds.				

Forest Loop 2		30 fps	856 x 480	3
A follow up to Forest Loop recorded on the same platform and route but including an initialisation motion at the start of each video designed to present ideal conditions for ORBSLAM, LSDSLAM and DSO to initialise. Each video is again a single completed loop, and lasts approximately 6 minutes and 30 seconds. The difference between the videos is primarily that light levels are reducing between recordings. For this reason, video 1 is labelled ”Ideal” and used for the majority of experiments in Chapter 5.				
SFU Forest		30 fps	752 x 480	4
Recorded from an all terrain vehicle travelling a route along forested mountain paths (Bruce et al., 2015). These were split in half into a road section (below) and offroad forest section that followed. The same route was followed four times under different environmental conditions: dry, wet, dusk and night. The route involves the vehicle turning and retracing its initial path.				
SFU Road		30 fps	752 x 480	4
The first half of each of the SFU videos used for SFU Forest (Bruce et al., 2015). These routes follow a mountain road, with trees lining either side, and are lacking much visual variety.				
Bebop Indoor		30 fps	1920 x 1080	1

<table>
<tr><td colspan="5">Recorded from the same Bebop drone as Bebop Forest and Loop, but in an office environment to provide a direct comparison. Performs a loop of the office and then turns and retraces that loop, detouring part way to briefly leave into a corridor. 2 minutes and 13 seconds long.</td></tr>
<tr><td>TUM Indoor</td><td>  </td><td>50 fps</td><td>1280 x 1024</td><td>10</td></tr>
<tr><td colspan="5">Dataset recorded from a hand held camera exploring office space (Engel et al., 2016b).</td></tr>
<tr><td>TUM Outdoor</td><td>  </td><td>30 fps</td><td>1280 x 1024</td><td>6</td></tr>
<tr><td colspan="5">Dataset recorded from a hand held camera exploring built up outdoor space, such as around buildings and through car parks (Engel et al., 2016b).</td></tr>
<tr><td>KITTI</td><td>  </td><td>10 fps</td><td>1241 x 376</td><td>10</td></tr>
<tr><td colspan="5">Dataset recorded from the roof of a vehicle driving along roads in a built up area (Geiger et al., 2012)</td></tr>
<tr><td>Virtual KITTI</td><td>  </td><td>10 fps</td><td>1241 x 376</td><td>5</td></tr>
<tr><td colspan="5">Dataset designed as a simulated replica of some of the videos from KITTI (Gaidon et al., 2016).</td></tr>
<tr><td>Unreal</td><td>  </td><td>30 fps</td><td>640 x 480</td><td>5</td></tr>
</table>

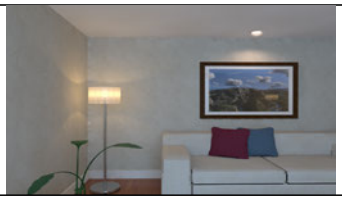
<table>
<tr><td colspan="5">Dataset recorded from routes around an Unreal Engine simulation of a forest (GmbH, 2022). Different starting locations and flight patterns, mostly straight lines. Motion of the camera is unnaturally smooth compared to flown or hand held. 4 videos each roughly 1 minute long and 1 video of 6 seconds only containing a strafing calibration flight.</td></tr>
<tr><td>Unreal Traits</td><td>  </td><td>30 fps</td><td>856 x 480</td><td>16</td></tr>
<tr><td colspan="5">Dataset recorded from an upgraded Unreal simulation in Chapter 6, with videos of the same route, a single loop, flown in controlled environmental conditions, such as wind, shadow and snow. There are 9 of these in total, each approximately 1 minute long and covering 54m (though distance in simulation is only partially meaningful). Also contains stationary camera versions, 7 x 1 minute.</td></tr>
<tr><td>ICL NUIM</td><td>  </td><td>30 fps</td><td>640 x 480</td><td>4</td></tr>
<tr><td colspan="5">Dataset of routes around a simulated living room (Handa et al., 2014). Provides a ground truth not just for camera trajectory but also for the 3D construction of the environment, enabling more in depth evaluation.</td></tr>
</table>

Table 3.1: Summary of all datasets reported within this thesis.

Table 3.2 is presented as a summary of where datasets appear in this thesis.

Dataset	Chapters
Hillwood AR	4, 5, 6, 7, 8
Hillwood Bebop	4, 5, 6, 7, 8
Forest Loop	5, 6, 7, 8
Forest Loop 2	5

SFU Forest	4, 5, 6, 7, 8
SFU Road	4, 5, 6, 7, 8
Bebop Indoor	4
TUM Indoor	4, 5, 6, 7
TUM Outdoor	4, 5, 6, 7
KITTI	4, 5, 6, 7
Virtual KITTI	4
Unreal	4, 5, 6, 7, 8
Unreal Traits	6
ICL NUIM	7

Table 3.2: Summary of which chapters each dataset appears in.

3.2 SLAM System Setup

3.2.1 SLAM System Versions

System	Commit	Source
ORBSLAM	f2e6f51	https://github.com/raulmur/ORB_SLAM2
LSDSLAM	d1e6f0e	https://github.com/tum-vision/lsd_slam
SVO	d616106	https://github.com/uzh-rpg/rpg_svo
DSO	ae1d0b3	https://github.com/JakobEngel/dso

Table 3.3: Sources and commit versions for each of the SLAM systems used.

State of the Art monocular SLAM systems, as highlighted in Chapter 2, were sourced from publicly available repositories provided by their original authors, as shown in Table 3.3. We initially chose four well-cited systems that would all run within the same test environment, and where necessary in later experiments we narrowed our

focus.

3.2.2 Test Machine Specification

All of the SLAM systems tested here were run on the same machine, with the following specification:

- Processor: Intel Core i7-4710MQ CPU(2.50GHz 8)
- Graphics: Intel Haswell Mobile
- Memory: 8GB
- Storage: 250GB
- Operating System: Ubuntu 14.04

As the use case for these SLAM systems is for them to be able to run live onboard a drone with minimal computational capacity, the specification of this test setup is a reflection of that.

3.2.3 Stochasticity

A variety of factors mean that the performance of a given SLAM system on a given dataset is not identical each time it is run, and therefore to properly test performance repeated trials are required. The reasons for this include randomness in the proposal, acceptance, or pruning of features (for example in the method ORBSLAM uses to limit the number of features within small areas), or in the interaction between the parallelised threads of different subsystems (for example where some other system process on the test platform requires increased resource and the SLAM system is left with less for one of its threads). To partially limit the impact of this, aside from using the same test computer every time we also make sure no other major applications are running at the same time and perform all repetitions of a particular experiment in the same session.

3.2.4 Parameters

These systems have similarities and differences in terms of the choice of parameters, as well as the guidance provide for tuning. ORBSLAM, LSDSLAM and SVO each have around a single page of parameters, while DSO has four times as many. ORBSLAM and LSDSLAM provide configuration files highlighting key useful parameters and allowing these to be easily swapped out for different setups. SVO sets its parameters via the ROS launch file, and for descriptions of those available we have to delve into the code. DSO's parameters are primarily within header files and require recompiling to change, with the authors even going so far as to recommend users don't modify "most" of them. All systems require camera calibration and distortion parameters be provided, although the exact format isn't the same in all cases. These are also separate from the header files for SVO and DSO, so they are the only parameters changeable without recompiling for DSO. The other parameters provided for each system will be discussed below, grouped by their function.

3.2.4.1 ORBSLAM

Camera: ORBSLAM takes two additional camera calibrations parameters in the form of FPS (frames per second) and RGB, which indicates the ordering of the colour channels. These need to be set correctly for the traits of the data, but there is no tuning to be done beyond that.

Features: nFeatures sets the number of features the ORB feature extractor aims to extract from each image. While 1000 is the default, and the authors recommend 2000 for high resolution images, we have found in preliminary experiments that going higher still, such as 3000, is more reliable in feature-rich environments.

Pyramid: Features are placed on corners chosen by a FAST corner extractor at a number of scale levels, which can be controlled by the nLevels parameter. The

resolution of each subsequent scale level is a factor of the previous determined by the scaleFactor parameter. As other work has shown (Milford, 2013), features extracted at lower resolution can be more resilient to low level motion, as they blur out small local structures (like individual leaves) and leave larger structures (like trunks). If we want more focus on lower resolution scales, this could be achieved either by adding more pyramid levels or by increasing the scale factor. The latter would be more efficient if we expect processing time spent on higher resolution levels to be less reliable.

Corners: FAST corners are thresholded on how different a circle of surrounding pixels are from the central pixel. Initially ORBSLAM splits the image into a grid and uses iniThFAST as the intensity threshold, but if it is unable to find enough corners in each grid square this way it will try again with minThFAST. Lower values here will detect corners in lower contrast images or environments, but also generally detect a larger number of less distinctive corners so for performance sake are to be avoided unless necessary.

3.2.4.2 LSDSLAM

Noise: Pixels must be at a point of sufficient gradient in the image (above threshold minUseGrad) to be included in mapping. In this way, LSDSLAM avoids using pixels in large, untextured areas where aliasing is a risk. If camera has a lot of noise, and therefore the chances of occasional texture in those flat regions increases, then this threshold should be increased. The cameraPixelNoise parameter is used as a measure of how noisy sensor data is, for reliability calculations. The authors recommend overestimating it.

Mapping: LSDSLAM exposes more control over what subcomponents are running at any time, starting with doSLAM, which turns mapping off entirely, leaving only odometry. The doMapping parameter, when disabled, turns off the

creation and updating of keyframes entirely so that the map stays static.

Keyframes: If you have limited memory and want the map to expand, but not in previously explored areas, then doKFReActivation means old keyframes are reactivated and updated instead of creating new ones, where possible. The frequency at which new keyframes are added the rest of the time is controlled by KFUsageWeight (higher for more visual overlap between frames) and KFDistWeight (higher for shorter distance between frames).

Loop Closure: useFabMap turns on or off the global loop closure functionality, while loop closures are always sought between local keyframes. The number of candidate matches considered for each new keyframe is controlled with maxLoopClosureCandidates, where a larger number will increase chances of finding matches but also increase computational load. loopclosureStrictness is a threshold on a consistency check between proposed loop closure keyframes before the closure is applied to the map. As this is a distance measure, a higher threshold will allow less similar (possibly incorrect) loop closures to be accepted.

Relocalisation: similar to loop closure detection, but when the frame to frame tracking is currently lost, the quality of a match between the current frame and an existing keyframe before it is considered good enough for relocalisation is controlled by relocalizationTH.

Depth Estimation: Pixel depths are calculated through stereo disparity. This can even be done at a sub-pixel level, at greater computational cost, if the useSubpixelStereo parameter is enabled. Some error in depth estimation is expected, and some pixels can end up with negative depth estimates, especially if their true depth is close to the image plane. Disabling allowNegativedepths can cull these obviously incorrect depth estimates, but at the cost of this being biased towards culling near-field pixels. If points close to the camera might be important to our application, then this parameter can be enabled instead. Finally,

depthSmoothingFactor sets how much depth estimations should be smoothed across adjacent pixels, with a lower number being (perhaps unintuitively) smoother.

3.2.4.3 SVO

Full descriptions available in svo/include/svo/config.h, but the function and variable names are not formatted the same here so anyone intending on using it for reference should focus on the variable name (e.g. after getInstance()).

Files: the initial parameters are just used for defining file names and locations.

- **Pyramid:** As with ORBSLAM, n_pyr_levels is the number of scale levels in the pyramid of images corners are detected from.

- **Keyframes:** SVO maintains a core of keyframes which form the backbone of the map and are used for bundle adjustment. Setting core_n_kfs higher will allow effective global optimisation over larger maps, but also increase computational complexity. Similarly, the total number of keyframes can be capped by using max_n_kfs if needed to control size. When selecting new keyframes, a minimum distance (relative to the height in the map) is applied from kfselect_mindist, so this can also be set higher to space keyframes out.

  IMU: use_imu can be set True if your dataset includes IMU data that you want to integrate. The delay between the camera and the IMU is then defined by img_imu_delay.

- **Initialization:** When the system initialises based on the first keyframes, several checks are applied to ensure a good initialisation. These require a minimum disparity (init_min_disparity), minimum number of tracked features (init_min_tracked) and minimum number of inlier features after RANSAC has been performed (init_min_inliers). If initialisation is failing, lowering these requirements might

help. Conversely, if initialisation succeeds but the resulting tracking is of poor quality, then these values could be increased.

Tracking: Map points are projected into each camera frame by SVO as a way of refining the tracking estimation. Lucas Kanade is used as the solver, and the minimum and maximum acceptable results are defined by klt_min_level and klt_max_level. The threshold on the quality of the resulting reprojection is thresholded by reproj_thresh.

Optimisation: Bundle adjustment of the camera pose is performed after the initial reprojection, for a defined number of iterations (poseoptim_num_iter) and thresholded (poseoptim_thresh), followed by bundle adjustment of the structure of the 3D points (structureoptim_num_iter) for a limited number of points (structureoptim_max_pts) and finally bundle adjustment of both local keyframes and 3D points together (loba_num_iter) using a Huber kernel with parameterised threshold (loba_thresh).

Corners: SVO attempts to place features on Harris corners. To control how strong a corner response has to be to be passed on for triangulation, set triang_min_corner_score.

Features: The cap on the maximum number of features being tracked (rather than in the map) is defined by max_fts, similarly to ORBSLAM. The quality of the tracking is also measured by whether the number of features drops below a threshold (quality_min_fts) or if the number of features lost within a single frame is above a threshold (quality_max_drop_fts).

3.2.4.4 DSO

The full parameter list is provided in settings.cpp, and the authors' advice that most of these should not be touched is emphasised by the need to recompile any time you want to change one. A smaller set of parameters are exposed through the user interface

at runtime. The first half of these, from the "mode" slider down through all of the checkboxes to "showCoarseTracking", and the first four sliders (sparsity, relVarTH, absVarTH, minRelativeBS) are controls on what outputs are displayed and how, and so not relevant to tuning performance. The remaining five are outline below.

activePoints: this sets the total number of features (new and those continuing to be tracked) the system is aiming to maintain in the active window. If increasing pointCandidates (below) to deal with heavy texture, you might similarly want to increase this. Whether this is necessary will depend on how good DSO is at quickly narrowing down candidate features to only reliable ones. If it does this well, only increasing the number of candidates will be necessary in scenes with a lot of texture.

pointCandidates: as with nFeatures in ORBSLAM, this sets the number of new feature points to consider with each new frame. Similarly with other systems, we would expect increasing this to help in a feature-rich environment, but also cost in processing.

maxFrames: defines a cap on the number of key frames that can make up the current active window (the key frames still in view). When a new key frame is proposed beyond this limit, an older frame will me marginalised out of the active window along with those frames which are not in sight. Having a larger maximum number of frames therefore keeps more key frames in the active window for longer, and allows for more time to locally optimise and match points against each other, at the expense of memory and processing costs.

kfFrequency: a scaling factor on the number of key frames taken (e.g. setting this to 2 will double the rate of new key frames being added).

minGradAdd: sets the minimal gradient value, similar to minUseGrad in LSD-SLAM, for a pixel to be included in the map. Similarly, setting this lower will

allow fainter edges to be included, but also increases the risk of including non edge pixels by mistake.

3.3 Modular Vision Framework

Much of the work presented within this thesis has involved the construction, variation and evaluation of chains of visual processing steps. Anticipating this would be the case, initial work was put into constructing a framework for visual processing modules, easily combined and rearranged for experiments without the need to write new scripts from scratch each time. This system was named EnVisionEd.

EnVisionEd is written in Python 3.0 and is designed to have minimal dependencies to allow for portability and ease of use. No other packages are required for the basic scaffolding functionality, though most of the existent modules require OpenCV (for most of the visual processing), NumPy (for matrix manipulations) and Matplotlib (for graphing). The place matching modules created for Chapter 8 have more complex dependencies due to their use of neural networks, and are an example of why it is good to have each module's dependencies decoupled from the others; EnVisionEd can be utilised fully without these dependencies as long as the place matching module is not called.

3.3.1 Design and Architecture

Enabling rapid prototyping of different processing sequences produces two main design priorities: 1) processing steps should be broken up into atomic parts that can be mixed around without having to handle dependencies between them and 2) any given prototype sequence should be easily definable with minimal knowledge of the workings of the system. This design addresses both priorities.

The highest level component in our framework is the System class, which handles collecting and interfacing the set of modules being used together in a particular in-

stance. System contains a dictionary called Data, which is passed to each module in turn as they are called. A module knows what keys to query the dictionary for, in order to get its required inputs, and likewise what keys to store its outputs under. In this way, modules can be agnostic of each other, only caring whether a module earlier in the processing chain has provided the right data for it and not needing to know what the module is or what it does. Defining a System can therefore be abstracted to a list of modules, in the order that they should be run, and often no additional fine tuning is required to integrate those modules with each other.

The method for defining which modules are to be used by a System takes a comma separated list, then searches the local environment for modules with the given names, allowing System itself to also be agnostic to the existence of other modules. If a module has any parameters, these are passed in brackets after the name. In more complex cases where multiple copies of the same module are required, modules can be given a label and the module constructors can be passed alternative input and output keys (provided in brackets in the comma separated list), so for example a system with two sources of camera data can store one of these under "Camera2" and the relevant other modules can be told to look here instead of the default "Camera".

Modules are split into four main types by their general use: Sensors, Processors, Displayers and Reporters. Defining a new module is simply a matter of inheriting from one or more of these classes. A System accepts four separate definition strings, one for each of these types of module, and they are handled slightly differently. When run, a System will loop over its set of modules in their defined order, starting with Sensors, then Processors, then Displays, calling a method on each which varies depending on their type (making it clear what a module is being called for, if it has multiple types). The System continues until either a module sets the exit flag in Data, or its Sensors stop running. At this point, Reporters will be run once each. More information on each module type follows:

Sensors are responsible for reading in data to the system. They typically have no depen-

dencies on other modules and therefore do not read from Data, but do write to it. The source of read data will usually be files on the local system, but it could also be a live feed such as a webcam. Sensor modules are called first at the start of every loop, but they can be implemented to preload all of their data when they are initialised. The System also polls Sensors each iteration for their "running" status, and if they have stopped the System will terminate.

Processors perform operations on existing data. They typically depend on inputs already in Data and will also typically write their results to it, though it is possible to make a Processor module that is for example counting and therefore requires no input. The System calls Processors after Sensors each iteration, in the order they were defined.

Displayers are modules which read from existing data but do not write anything. They will take keyed values from Data in order to produce an output from them, such as a print out to the command line or an image window. These are modules which are not necessary for any system's performance, allowing them to be turned off if the user does not need to see outputs while the system runs. They are called at the end of each iteration of the System.

Reporters are similar to Displayers in that they read and produce outputs, but do not save their results back into Data. Reporters are called once, at the end of a System's run, rather than every iteration. They are typically used for saving final results to files or displaying graphs of them.

As a helpful tool for running experiments with multiple systems and datasets, EnVisionEd provides the Evaluator class. Evaluators can be given a list of already defined Systems and datasets, then will run through the Systems in turn, setting each dataset as the input and running the System to completion before moving on to the next. Also included is a sanity check utility that automatically runs through all of the defined

modules for a System, querying their input and output keys, and outputs all of the dependencies, along with a warning if any module appears to be lacking a required input. There are still cases where this cannot confirm the validity of a defined System, but it is certainly a useful tool for picking up on many common mistakes.

The below simple example script demonstrates the core functionality in practice. A new System is created, and four text strings defining the modules it should use are provided to it. In this case, two greyscale images are created, one from the blue channel of the camera feed and one from the green. They are then normalised and displayed to the user. A framerate module is called as a Processor each iteration to track performance, and then called once more at the end as a Reporter to provide a summary.

```
sys = System(True)
sensors = "Camera:CameraFeed(1)"
processors = "TimeKeeper," \
             "G1:BGRtoGrey(Camera;Grey1;True;False;False)," \
             "N1:Normaliser(Grey1;Grey1;1)," \
             "G2:BGRtoGrey(Camera;Grey2;False;True;False)," \
             "N2:Normaliser(Grey2;Grey2;1)," \
             "TimeKeeper,␣Framerate"
displayers = "CameraDisplayer,N1,N2"
reporters = "Framerate"
sys.define(sensors, processors, displayers, reporters)
sys.start()
```

Simple example script creating a processing system with EnVisionEd

Chapter 4

Monocular SLAM in Forest Environments

4.1 Introduction

As discussed in the previous chapters, Monocular SLAM approaches are primarily developed and evaluated in structured, man-made environments such as offices and urban roads. The lack of variety in these test environments introduces the potential risk that existing state of the art approaches are unintentionally adapted to specific conditions.

This chapter presents our first contribution, in which we perform an in depth analysis of environmental factors known to cause difficulty for Monocular SLAM systems, and how they vary between traditional man-made environments and the natural conditions found in forests.

We compare the performance of state of the art Monocular SLAM systems ORB-SLAM, LSDSLAM, DSO and SVO on datasets including both existing and newly collected forest videos. LSDSLAM and SVO fail to initialise on any dataset, while ORBSLAM and DSO initialise on a single video but still demonstrate drift indicative of poor tracking and loop closure performance.

The main contribution of this work is in using statistical measures which highlight key visual differences between the video datasets and help to confirm the presence of challenging environmental conditions: rapid lighting changes, in-scene motion and heavy texture. The statistics used to measure these conditions are primarily inspired by visual ecology literature, where they are used to describe environments for the purpose of investigating how animal visual systems are adapted to them.

Our final contribution is the development of a photorealistic forest simulation environment. We test the state of the art Monocular SLAM on videos from this environment as well, but crucially we also compare the scene statistics of scenes from the simulation with those of real forests. The stark difference between simulation and real in this comparison is taken as a warning against relying on simulators for the development of new forest-capable SLAM approaches.

4.2 Experimental Methodology

The data collection for the creation of the Hillwood datasets used in this work is outlined in Methodology (Chapter 3). The SLAM system versions used are those described in Chapter 7.2.4. All other scene stats experiments were run using our modular vision system (Chapter 3.3), with a setup described below. Details of what each module does can be found in Appendix A.

Three main test scripts perform the experiments for this work: “SceneStatsICRA” (for Luminance, Contrast and KLD), “LaplacianICRA” and “CoVisICRA”, saved as these paper-specific versions in case the process was updated later. These follow the same general layout as each other but computing the Variance of the Laplacian involves a different set of preprocessing steps from the other scene statistics and computing the covisibility of features requires none at all. The processing of SceneStatsICRA is also considerably faster, and so iterating on these in isolation during development sped up the task.

Depending on the format of the dataset, it is loaded with either a CameraFeed, ROSFeed, or ZipFeed module. CoVis does not use any preprocessing modules. For LaplacianICRA, we preprocess with Greyscaler, Resizer, Normaliser and IntensityHist, while for SceneStatsICRA, it is just GreyScaler and Normaliser. FPS correction, which is done only for SceneStatsICRA, is done with the "set_rate" function function of the test System we are constructing. In the SceneStatsICRA case, we actually run two copies of Normaliser (one in the range 0 to 255 and one in the range 0 to 1.0), because the image processors need the latter format and the module to create intensity histograms, as well as the ORB feature extractor, require the former.

The main processing modules for SceneStatsICRA are Luminance, Contrast, KLDivergence, along with a Difference module for each of Luminance and Contrast. All five of these modules then also have a Grapher module, which inherits from Lister and therefore keeps track of all of the values for the whole run, while allowing live Graphing to debug. Finally a FileReporter is added as a reporter module to save all five of the resulting lists to files. For LaplacianICRA we use only Laplacian, a single Grapher for it, and FileReporter. CoVisICRA requires FeatureExtractor, FeatureMatcher, two Counter modules (one for Keypoints, one for Matches), a Percentage module running off of the two Counters, and a Lister (as this test runs too slowly for us to feel the need to live graph it), again being recorded by FileReporter.

For analysis, and presentation here, saved results are later loaded and displayed by BoxPlotter.

4.3 Publication: Visual Appearance Analysis of Forest Scenes for Monocular SLAM

This work was published as a contributed paper in the proceedings of the IEEE International Conference on Robotics and Automation (Garforth and Webb, 2019).

This is all a personal contribution, with the exception of the guidance and editorial

of written content provided by Prof Barbara Webb.

Visual Appearance Analysis of Forest Scenes for Monocular SLAM

James Garforth[1] and Barbara Webb[1]

Abstract— **Monocular simultaneous localisation and mapping (SLAM) is a cheap and energy efficient way to enable Unmanned Aerial Vehicles (UAVs) to safely navigate managed forests and gather data crucial for monitoring tree health. SLAM research, however, has mostly been conducted in structured human environments, and as such is poorly adapted to unstructured forests. In this paper, we compare the performance of state of the art monocular SLAM systems on forest data and use visual appearance statistics to characterise the differences between forest and other environments, including a photorealistic simulated forest. We find that SLAM systems struggle with all but the most straightforward forest terrain. We identify key attributes (lighting changes and in-scene motion) which distinguish forest scenes from "classic" urban datasets. These differences offer an insight into what makes forests harder to map and open the way for targeted improvements. We also demonstrate that even simulations that look impressive to the human eye can fail to properly reflect the difficult attributes of the environment they simulate, and provide suggestions for more closely mimicking natural scenes.**

I. INTRODUCTION

This paper looks at the problem of performing visual Simultaneous Localisation and Mapping (SLAM) in unstructured natural environments (such as forests), rather than the structured man-made environments (such as offices and city streets) that play host to the majority of SLAM research [1], [2], [3]. Our target application is forestry, where data gathering to assess tree health could be greatly enhanced in efficiency, scale and accuracy if robots could navigate within forests. Unmanned aerial vehicles (UAVs) have the agility to traverse uneven or vegetation-cluttered terrain and inspect trees at any height, but in this scenario might not be able to carry many sensors, so our focus is on monocular SLAM. There are a number of reasons to expect that SLAM may be difficult in forests. Global Positioning System (GPS) data tends to be unreliable under canopy cover. Vegetation is locally dynamic due to wind and patchy light, yet has high global similarity that could lead to substantial aliasing. The ability of state of the art SLAM algorithms to deal with such conditions is largely untested. Our main contributions are:

- Qualitative analysis of the performance of monocular SLAM algorithms in forest environments.
- Characterisation of fundamental visual differences between forests and more traditionally mapped environments (e.g. offices) using scene statistics.
- Assessment of photorealistic simulation as a ground-truthing environment for developing SLAM algorithms.

*This work was supported by the Edinburgh Centre for Robotics and the Engineering and Physical Sciences Research Council.

[1]James Garforth and Barbara Webb are with the School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh, EH8 9AB, United Kingdom. {

Fig. 1: Our photorealistic forest environment (Left), alongside a region of real forest from our Hillwood dataset.

II. BACKGROUND

A. Mapping Forests

Forest management is greatly aided by accurate measurement of trees, but below the canopy, few survey methods have been successfully automated. Humans with measuring tapes are still the standard method for collecting tree trunk diameter data, for example. Scanning lasers have been used to gather higher quality data but still require humans to walk the area to be mapped, carrying heavy equipment instead of a tape. This approach also requires substantial post-processing of point clouds. In [4] points are clustered into individual trees; aligned to an existing map, fine tuned by ICP, and diameters at a fixed height extracted. Pierzchala et al.[5] improve the performance of offline laser maps in forests with loop closure techniques from SLAM.

Miettinen et al.[6] developed a real-time laser-based system, mounted on a mobile platform, but found it very difficult to match individual trees in all but the sparsest managed forests. Ohman et al.[7] added cameras to this system, as the visual appearance of bark on tree trunks makes them easier to differentiate. They use a phase congruency edge detector to deal better with low contrast images, but were still unable to obtain usable levels of accuracy and processing speed.

The depth camera of Google's Tango device has been shown to provide sufficiently accurate reconstruction of forest scenes[8], but only on a small scale.

B. Monocular SLAM

A key advance in the field of monocular SLAM was Klein and Murray's Parallel Tracking and Mapping (PTAM) [9], which has been successfully used on UAVs [10], [11]. PTAM, as with most monocular SLAM systems after it, is a "keyframe" based method. Instead of storing sensor data from every single camera frame, PTAM selects a smaller set of frames that it judges most representative, and builds the map from these. Mapping only needs to occur when keyframes are added, so can be run in a separate thread from camera motion tracking, improving real-time performance.

"ORB-SLAM"[1] adds faster feature comparison and greater view invariance to PTAM by using ORB features[12], extracted once and used for the tracking, mapping and loop closure subsystems. Instead of extracting features, "direct" SLAM methods, e.g. "LSD-SLAM"[13], use a metric to subselect useful pixels and perform alignment between images by minimizing photometric error. The volume of pixels used varies from selecting all of them ("Dense" methods like DTAM[14]) through to sparsely selected pixels in methods like "SVO"[15] and "DSO"[16]. Yang et al.[17] note that direct methods are more robust than feature-based when scenes have low texture and provide a more complete reconstruction due to their use of more image information, but are vulnerable to camera properties and rapid lighting changes.

Recently, ORB-SLAM has been used on a Bebop UAV [18] for monitoring crops in plantations, combined with GPS and Inertial Measurement Unit (IMU) data for navigation. Smolyanskiy et al.[19] use DSO for obstacle detection on a UAV performing autonomous trail following in forests. Though neither addresses the full complexity of forest survey, they indicate that ORB-SLAM and DSO are good starting points for experimentation.

C. Adapting SLAM to Environment Properties

The application of monocular SLAM in forests may require adaptation of the algorithms to the characteristics of the visual environment. Previous researchers have looked at methods for basic classification of visual environments to support switching between SLAM methods [20] [21]. In more directly relevant work, Saeedi et al.[22] perform "Design Space Exploration", searching through the parameter space of a SLAM algorithm for a desired trade-off between accuracy and efficiency, given a high level description of the environment. They use Kullback-Leibler (KL) divergence, an information theoretic measure of how different two distributions (in this case, intensity histograms) are from one another, and posit that this one statistic neatly encompasses the variation of structure and motion within the scene. Research into the adaptation of animal vision to natural environments [23], [24] suggests additional relevant 'scene statistics' might be luminance, contrast and colour distributions. In particular, lighting-related effects are well known to impact SLAM[25], with potential solutions explored in [26] and [27].

III. METHODS

A. Methodological Issues

We wish to evaluate the performance of mapping algorithms in a forest environment, but several issues make this difficult to do rigorously. Evaluation of the accuracy of pose estimation requires a ground truth pose, usually obtained from accurate external sensors, such as a calibrated camera rig or, outdoors, GPS. Forest canopy cover, however, interferes with GPS signals. Placement of some other form of tracking sensors at ground level in a large area of interest is costly, and if visible would inadvertently provide beacons that influence the performance of the systems we want to test. Evaluating 3D reconstruction also requires ground truth, usually obtained either by working in an environment of known, rigid structure, or by using a more complex but comprehensive mapping solution than the one being assessed (for example a high powered laser scanner). However, mapping of large natural environments with such scanning systems remains a difficult problem[28] and as such does not guarantee an error free ground truth.

An alternative method for achieving pose and reconstruction ground truths is to perform experiments in a simulated environment where both are known perfectly. Simulation would also provide much greater control over key factors expected to affect performance, such as lighting and wind, allowing for comparison of different conditions. The trade-off is the possibility that a simulated forest does not properly capture the traits that make real forests challenging, and that improvements developed in simulation will not transfer to real world applications.

B. Datasets

We put together a selection of video datasets (summarised in Table I), including forest environments, structured urban environments and simulations, which would allow us to assess mapping algorithms and compare visual properties. The datasets come in a number of formats, such as video files, folders of individual frames or in the "rosbag" format used by the Robot Operating System[29].

The first of our forest data is from the SFU Mountain Dataset[30] (henceforth simply SFU), recorded from a wheeled robot driving a forested mountain road in a variety of weather and lighting conditions. We use only the "Dry" conditions to avoid our results also reflecting the effects of weather. Partway through each video, the vehicle moves from open road to a canopy-covered dirt track. As we are primarily interested in how the latter scenario differs from the former, we split the video at this point, forming "SFU Road" and "SFU Forest". We use the left of the vehicle's two forward facing cameras as a single video stream.

We also recorded our own forest videos, referred to here as Hillwood. This dataset contains videos recorded from two low cost UAV platforms (Parrot's AR.Drone and Bebop) and these are analysed separately. Hillwood provides more complex camera motion than SFU, as the camera takes winding routes and retraces the same area. It also contains sequences away from any path, over and around vegetation, to test performance of SLAM in the absence of any clear man-made structure such as a track.

We chose two "classic" datasets to represent the more structured environments typically used for testing SLAM applications: the TUM Monocular dataset[31], recorded on a hand held camera; and KITTI[32], recorded from a car on city streets. We further sample from TUM Monocular to make two datasets: one indoor (offices) and one outdoor (urban). For purpose of comparison, we also include Bebop Indoor, an office video recorded on the same platform as Hillwood Bebop.

Additionally, we recorded a simulated dataset utilising the Unreal Engine and a set of photorealistic forest assets

created for film and video game rendering[33], and manually flying the virtual camera through it. Although in principle this should resemble the Hillwood data (a drone travelling through unstructured forest) we note that in practice this produced much smoother motion than a real UAV.

C. SLAM System Comparison

We selected four state of the art monocular SLAM systems for comparison: (1) ORBSLAM2[1], a sparse feature-based method, (2) LSD-SLAM[13] and (3) DSO[16], which are both direct methods and (4) SVO[15], which is semi-direct, using direct methods for tracking and then using features later in its pipeline. We intended to also include OKVIS [34], but our chosen datasets do not provide the required inertial measurements or calibration data. We intended to assess the benefits of visual inertial SLAM in future.

The selected systems were all calibrated for and run on the Hillwood, SFU and Unreal datasets to assess their performance on forest and simulated forest data. Demonstration of the performance of these systems on traditional indoor and outdoor scenes can be found in each of their original papers.

D. Visual Appearance Metrics

Forest environments could have some global visual properties that differ from structured environments and thus set a challenge for visual SLAM. To assess this, we use these statistics to compare the datasets:

1) **Lighting changes** as we expect that the effect of the forest canopy will be frequent large switches between bright sunlight and shadow. We measure both luminance and contrast. Luminance is defined as the average intensity value of a frame. We use Root Mean Squared Contrast, defined as the standard deviation of luminance in a frame, divided by the mean. Change is recorded as the difference between each subsequent pair of frames for each of these statistics.

2) **Kullback-Leibler divergence** of intensity histograms, as an approximation of scene structure and motion[22]. We expect that the heavily deformable nature of forests (leaves, grasses etc.) leads to a large amount of motion in the scene with even a small amount of wind. KL Divergence for intensity images is calculated as in [22].

$$D_{KL}(I_t \parallel I_{t-1}) = \sum_{t=0}^{256} I_t(u) \log \frac{I_t(u)}{I_{t-1}(u)}$$

Where I_t and I_{t-1} indicate the normalised intensity histograms (256 bins) of the frames.

3) **Variance of the Laplacian** approximates the frequency and strength of edges within the image. In this way it can give us information about two traits of our datasets: firstly, how in focus the images are (as excessive camera motion will cause blurring) and secondly how complex the textures in the scene are. Implemented using OpenCV's Laplacian function, we also rescale all images to the same size (320x240) beforehand as image size effects the result.

We also looked at two "secondary" statistics, which help to exclude non-environment specific attributes of datasets from being the major factors in our results:

1) **Features Matches** are used as a simplified measure of the ease of tracking features for visual SLAM in the absence of ground truth data for our datasets. SIFT features are extracted and matches sought between subsequent pairs of frames (100 features per frame). We use a brute force matcher, then a ratio test to decide which matches to accept.

2) **Reprojected Similarity** estimates frame to frame overlap in order rule out the possibility that large differences in the primary statistics are due to large camera motions. The similarity is calculated as the Mean Squared Error of two subsequent frames after using feature matches to reproject them into the same frame of reference.

These measures were implemented in Python using OpenCV. We calculate all of the change statistics for subsequent pairs of frames, sampling the datasets at the same frame rate (10fps) to account for the expectation that a camera sampling faster will see smaller changes between frames. Our pipeline converts images to intensity (grey) and normalises before it calculates the statistics. It is also worth noting that we skip the first 30 frames of each video because in some datasets these contain artefacts from the camera's auto-calibration which can skew the results.

IV. RESULTS

A. SLAM System Performance

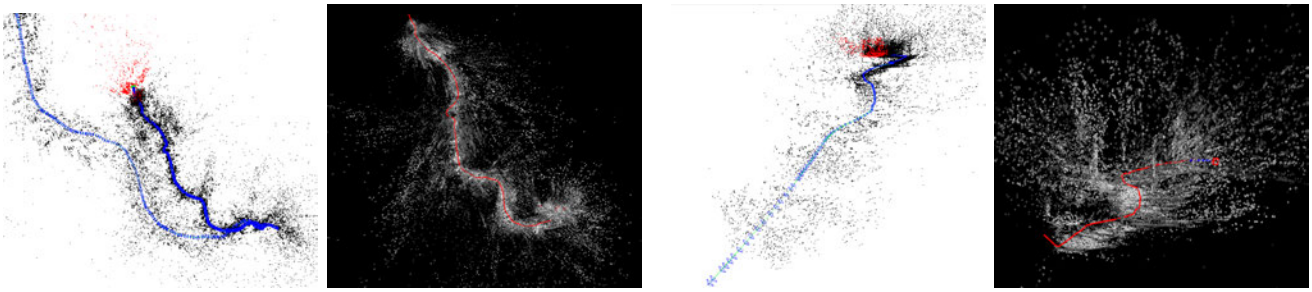
We tested four SLAM methods on the forest datasets: Hillwood AR and Bebop and SFU Forest, as well as the Unreal dataset. The only fully tracked sequences were obtained using either ORBSLAM2 or DSO on the SFU video (see Figure 2a), or on the simulated data. When run on Hillwood data, ORBSLAM2 and DSO achieved tracking for a small portion of the video (less than 1 minute), resulting in very small sections of map that we could not confirm the quality of by eye. LSD-SLAM and SVO fail to start tracking on any of the forest videos, either producing no map or one in which features are distributed with no recognisable structure.

For both ORBSLAM2 and DSO, the tracked pose drifts in scale over time, as is most notable from the misalignment of the outbound and inbound tracks. This is not unexpected from monocular SLAM systems, but the fact that neither system manages to recognise previously visited locations and correct the drift demonstrates a failure of their respective place recognition mechanisms. We note that the demonstrated tracking from these two systems shows there is no specific advantage in this environment for feature-based (ORBSLAM2) vs. direct (DSO) methods. Looking at the point clouds produced by the two systems in Figure 2b, however, we can see that the larger volume of data afforded by DSO's direct method does provide a more detailed reconstruction.

The failure of two of our four evaluated systems to even initialise pose tracking on any of the tested datasets demonstrates the difficulty of forest environments for even short term mapping. The two systems that were able to maintain tracking only did so on the SFU Forest video, and only if we substantially increased the number of features extracted per image (on ORBSLAM2 we use 3000 vs. the

TABLE I: A summary of the datasets reviewed, with video count, frame rate, resolution and a representative example frame.

Name	Hillwood AR	Hillwood Bebop	SFU Forest	SFU Road	Bebop Indoor	TUM Indoor	TUM Outdoor	KITTI	Unreal
Count	3	1	1	1	1	10	6	10	5
FPS	15	30	30	30	30	50	30	10	30
Res	320x240	1920x1080	752x480	752x480	1920x1080	1280x1024	1280x1024	1241x376	640x480
Eg.									



(a) Top-down views exhibiting the recovered route. Both track the outbound route well, but DSO struggles more with the return route.

(b) Horizontal views exhibiting the level of reconstruction. DSO's denser map recovers notably more structure.

Fig. 2: The full tracks and point clouds as produced by ORBSLAM2 (white background) and DSO (black background) on the forest video where they are most successful (SFU forest under dry conditions).

default 1000), suggesting that in this environment a large number of candidate features need to be extracted at each frame in order to ensure sufficient crossover between frames for matching.

The successful SFU video represents an easy use case for monocular SLAM, as the camera is mounted to a slowly moving ground vehicle and experiences very little roll, pitch or even yaw and all viewpoint changes happen gradually. The vehicle also sticks to a clear, well-defined dirt path, and observation of the point cloud produced by ORBSLAM2 especially (Figure 2a) seems to suggest that the ground here is providing the majority of the tracked features. Similarly, the Unreal videos seem to be less difficult for these SLAM systems, but as we discuss in the next section they are perhaps failing to reflect the difficulties of forest scenes.

In Hillwood videos, where all algorithms struggle, the camera was mounted on a UAV which follows a less straightforward route with more rotational motion in all axes. But tracking failures occur even on relatively straight sections, suggesting camera motion is not the only factor causing problems. Notably, these videos contain a lot less clear ground than SFU, instead often travelling over vegetation or fallen leaves. The UAV also frequently flies near to and between trees, leading to regular occlusions of parts of the scene, while the ground robot in SFU usually keeps enough distance from trees that this effect would be greatly reduced.

B. Visual Appearance Comparison

In the previous section we found that all the SLAM systems tested failed on the more challenging forest data, yet they have previously been established as effective systems in more typical scenarios such as indoor mapping and city roads. Hence, we did an initial investigation of whether there are any general differences in the scene statistics of these different scenarios (see methods).

1) Lighting Changes: Luminance (Figure 3a) and contrast (Figure 3b) changes do appear to differentiate between the forest datasets (Hillwood, SFU Forest) and most of the classic ones (TUM). The median differences for luminance and contrast are higher in the forest videos, irrespective of the platform they were recorded from, and the distribution is also larger, reflecting a tendency of these datasets towards both generally larger lighting changes over time and larger sudden lighting changes. KITTI is the only non forest dataset to see a similar distribution of lighting changes, but it is also the only dataset where the camera is travelling faster than walking speed. This large speed difference could account for the large visual appearance changes between frames.

When comparing Hillwood Bebop and Bebop Indoor the camera parameters are identical but the results differ, each following the general trend for the other datasets in their respective environment. Likewise, when we compare SFU Forest and SFU Road, where possible complicating factors such as time of day or camera motion and parameters are ruled out, we still see major lighting differences, supporting the possibility that the forest canopy is responsible for the lighting variability.

The Unreal dataset demonstrates the lowest median and the tightest distribution of values for luminance and contrast changes, likely caused by the simulator's lighting model. Even though the simulation is lit by directional lighting from an artificial sun, it also has ambient lighting. As a result, the areas of the simulated forest that are in shadow still appear relatively well lit, meaning the camera is less likely to experience large swings from light to dark as it transitions between direct sunlight and shadow.

2) Kullback-Leibler Divergence: It is clear from the KL divergence results (Figure 3c) that the datasets in forested areas (including SFU Road, which is lined with trees) are less

predictable frame to frame than classic environment datasets are. The difference between the two SFU datasets indicates that going off road and under canopy with the same platform markedly reduces the predictability further. All of this goes to support the idea that vegetation has a notable impact on how much scenes change over time, perhaps due to the amount of small-scale motion (e.g. of leaves) they introduce. The higher KLD values for the Bebop Hillwood data suggest that this is not the only factor, however. The other videos mostly travel forwards along a path, while the Bebop goes back and forth over one area, so it is very likely that the large amount of rotational motion by the Bebop also contributes to the KL divergence being considerably higher.

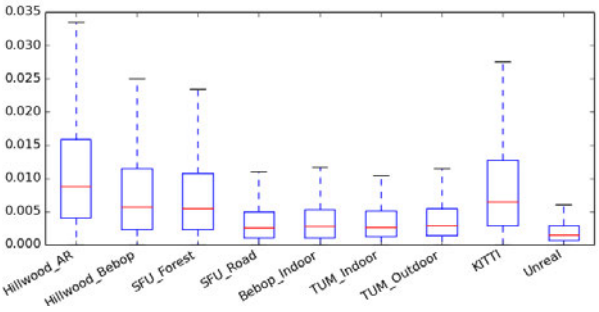
The very low KL divergence seen in the Unreal simulated data implies a high predictability of each frame given the previous one. It is not obvious to the human eye that this environment contains any less motion or complex structure than the real forest, in a way that would account for less predictability. We suspect that the difference may be in the camera model. The simulated camera does not suffer from any noise and applies considerable blurring, both of which serve to improve the similarity between subsequent images.

3) Variance of the Laplacian: The results for the variance of the Laplacian correlate well with the observed complexity in the datasets, such that the texture-heavy natural scenes sit at one end of the spectrum and indoor scenes (with flat textureless regions like walls) are at the other. A potential limitation of this measure is demonstrated by SFU Road, however, as the prominence of a large featureless sky throughout this video is the cause of its low median variance (confirmed by rerunning the pipeline with the top half of the video cropped). It is interesting to note that the high texture datasets are the ones that are more difficult for visual SLAM, as this typically benefits from textured scenes for feature extraction. It is likely that this visual complexity overwhelms feature extraction and matching and would explain why getting ORBSLAM2 to work required us to increase the number of candidate features extracted.

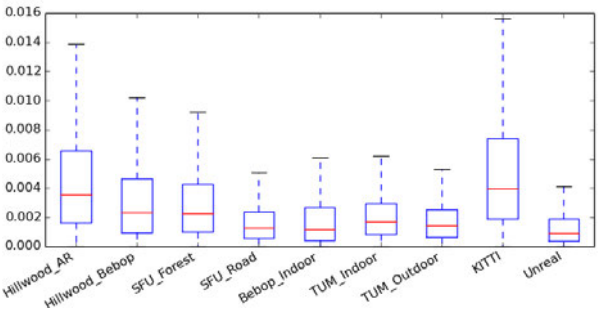
There is little evidence in these results to suggest that blur (which would reduce the variance) fis a problem in any of the real world data, as the datasets at most risk should have been those with a less stable camera (such as Hillwood, which reports the highest variances). The low variance for the simulated Unreal data, is likely caused by the game engine adding too much motion blur, anti-aliasing, or using a limited colour palette.

4) Feature Matches: The distinction between forest and classic datasets is less clear with respect to the percentage of good frame to frame feature matches, but there does still appear to be some trend. Both Hillwood datasets and SFU Forest achieve fewer median successful matches between frames, as well as having lower minimum matches. These results would imply that it is indeed harder to extract and continue to track features reliably from forest scenes than urban ones.

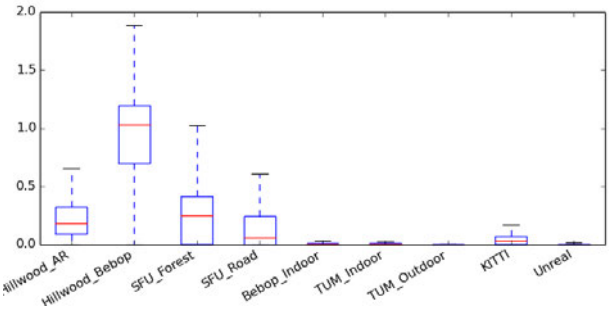
Similarly to the results with lighting changes, it makes sense that KITTI's lower results here would be caused by



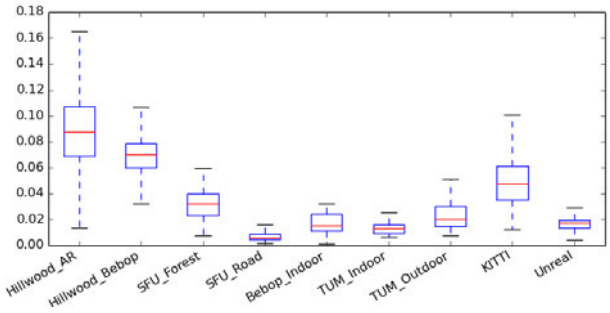
(a) Luminance changes between subsequent frames. Higher median and maximum changes for the three datasets under forest canopy on the left, but lower for the simulated Unreal forest.



(b) Contrast changes between subsequent frames. Higher median and maximum changes for the three datasets under forest canopy on the left, but lower for the simulated Unreal forest.



(c) Kullback-Leibler divergence between subsequent frames. Higher median and maximum changes for the four vegetation heavy datasets on the left, but not for the simulated Unreal forest.



(d) Variance of the Laplacian for all frames. The higher variance for outdoor (especially forested) datasets suggests the presence of more and stronger edges/texture.

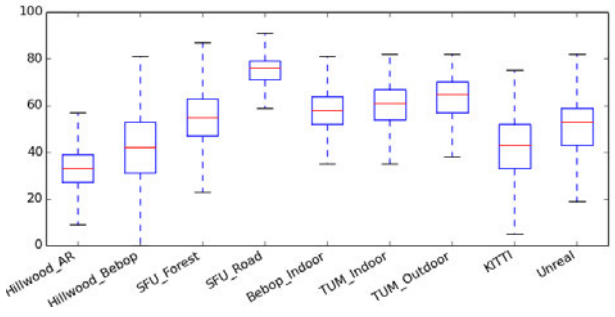
Fig. 3: Our primary statistics characterise differences between video datasets gathered in forest and urban environments.

the camera moving further between frames.

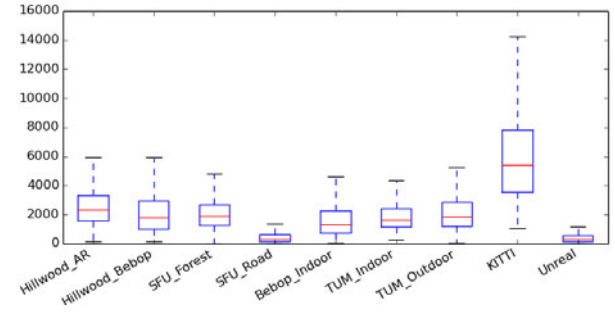
SFU Road achieves consistently high matches. As the skyline is always visible in this data, and has been shown to be useful for navigation[35], it is expected that this is providing a large number of reliable features. To test this, we reran the pipeline with the top half of each image removed, and found that the results for SFU Road did indeed become more like the other datasets.

Notably, this statistic is also the only one of those tested where the Unreal data does not stand out significantly from the real forest scenes.

5) Reprojected Similarity: After using matched features to reproject subsequent pairs of images into the same frame of reference, we see very little difference between most of the datasets. Seeing similar levels of overlap between frames in these datasets helps reject the idea that the other results reported here (primarily for KLD) could be caused by significant rotational or translational motion specific to the forest datasets, rather than an attribute of the environment. KITTI displays a much higher error after reprojection, as expected for a dataset where the camera is moving significantly faster and overlap would be expected to be smaller. SFU Road and our simulation, however, have notably lower errors than other datasets despite not being notably slower. The explanation is likely similar to IV-B.3 in that the sky, or the game engine's limited colour palette, result in self-similarity across the environment.



(a) Frame to frame SIFT feature matches, after ratio test, out of 100. Less reliable matching for forest datasets supports our observations of poor SLAM performance.



(b) Frame to frame overlap, as measured by Mean Squared Error after reprojection of each frame into the next. There is no notable difference between the overlap of frames in forest and non forest datasets.

Fig. 4: Secondary statistics, used as support for other claims rather than to directly characterise environments.

V. CONCLUSION AND FUTURE WORK

In this paper we evaluated the performance of state of the art monocular SLAM in forests. Such unstructured natural environments have not been traditionally considered in SLAM evaluations, despite the high potential for robot applications in this domain. We found that even with tuning of the parameters, only two systems (ORBSLAM2 and DSO) successfully ran, and only on the easiest of our test cases: a slow and steady ground robot travelling along a clear path in a forest. In most cases, systems failed to produce a usable map. These tests identify the unsuitability of any existing solutions for off the shelf use in this domain. One particular problem area observed is loop closure, which we hope to address with place recognition Convolutional Neural Networks in future work.

Improving the performance of existing systems requires an understanding of how the forest environment differs from the standard use-cases. We performed statistical analyses of forest and non-forest data and found some key differences. Lighting (represented by luminance and contrast) changes over time distinguish forests from offices and roads and are likely caused by the gaps and movement of the canopy leading to frequent variation in the amount of sunlight illuminating the scene. In-scene motion (represented by entropy measure KL divergence) is also notably higher in forest scenes, likely due to the presence of wind and flexible vegetation. This suggests that the key developments needed (to extend ORBSLAM or DSO which worked in the simpler case, or in new SLAM algorithms) are methods to deal more robustly with lighting and scene dynamics. These are already active areas of interest but gain additional motivation from our analysis.

We also investigated the use of highly realistic game engine based simulation as an alternative to real world data when testing an improved SLAM system for natural environments. We note that such a simulation would be useful in a number of ways, providing ground truth data that is hard to match in the real world and also allowing fine tuned control over the exact variables (light and motion) that we want to control for. We issue a warning, however, against abandoning real world data too quickly, as our scene statistics mark out the simulated forest as more different in appearance from real forests than urban environments are.

Finally, rigorous evaluation of SLAM systems for forests requires more complete test data than simply video. If a solution can be found to ground-truthing in real forests, perhaps by careful synchronisation with lidar data, then this can be used to create a new forest dataset. Alternatively, improvements can be made to our existing simulation, for example through the addition of realistic sensor models, and our statistical approach can be used to establish if a greater resemblance to real forest data has been achieved.

ACKNOWLEDGMENT

This work was undertaken with advice and guidance from forest mapping specialists Carbomap (carbomap.com).

REFERENCES

[1] R. Mur-Artal, J. Montiel, and J. Tardos, "ORB-SLAM: a Versatile and Accurate Monocular SLAM System," *arXiv preprint arXiv:1502.00956*, 2015.

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

[3] R. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pp. 127–136, 2011.

[4] T. Takashi, A. Asuka, M. Toshihiko, K. Shuhei, S. Keiko, M. Mitsuhiro, T. Shuhei, N. Shuichi, M. Akiko, C. Yukihiro, S. Kouji, and H. Toru, "Forest 3D Mapping and Tree Sizes Measurement for Forest Management Based on Sensing Technology for Mobile Robots," *Springer Tracts in Advanced Robotics*, vol. 92, pp. 357–368, 2014.

[5] M. Pierzchała, P. Giguère, and R. Astrup, "Mapping forests using an unmanned ground vehicle with 3d lidar and graph-slam," *Computers and Electronics in Agriculture*, vol. 145, pp. 217–225, 2018.

[6] M. Miettinen, M. Ohman, A. Visala, and P. Forsman, "Simultaneous Localization and Mapping for Forest Harvesters," *Proceedings 2007 IEEE International Conference on Robotics and Automation*, no. April, pp. 517–522, 2007.

[7] M. Ohman and M. M. Kosti Kannas, Jaakko Jutila, Arto Visala and Pekka Forsman, "Tree Measurement and Simultaneous Localization and Mapping System for Forest Harvester," *Field and Service Robotics Springer Tracts in Advanced Robotics*, vol. 42, pp. 369–378, 2008.

[8] J. Tomaštík, Š. Saloň, D. Tunák, F. Chudỳ, and M. Kardoš, "Tango in forests–an initial experience of the use of the new google technology in connection with forest inventory tasks," *Computers and Electronics in Agriculture*, vol. 141, pp. 109–117, 2017.

[9] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, IEEE. IEEE, nov 2007, pp. 1–10.

[10] S. Weiss, M. W. Achtelik, S. Lynen, M. C. Achtelik, L. Kneip, M. Chli, and R. Siegwart, "Monocular Vision for Long-term Micro Aerial Vehicle State Estimation: A Compendium," *Journal of Field Robotics*, vol. 30, no. 5, pp. 803–831, sep 2013.

[11] D.-N. Ta, K. Ok, and F. Dellaert, "Monocular Parallel Tracking and Mapping with Odometry Fusion for MAV Navigation in Feature-lacking Environments," *Intelligent Robots and Systems (...*, 2013.

[12] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2564–2571, 2011.

[13] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," in *Computer Vision ECCV 2014*, ser. Lecture Notes in Computer Science, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, vol. 8690, pp. 834–849.

[14] R. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *2011 International Conference on Computer Vision*, IEEE. IEEE, nov 2011, pp. 2320–2327.

[15] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 15–22.

[16] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, 2017.

[17] N. Yang, R. Wang, and D. Cremers, "Feature-based or Direct: An Evaluation of Monocular Visual Odometry," pp. 1–12, 2017.

[18] V. Duggal, M. Sukhwani, K. Bipin, G. S. Reddy, and K. M. Krishna, "Plantation Monitoring and Yield Estimation using Autonomous Quadcopter for Precision Agriculture," 2016.

[19] N. Smolyanskiy, A. Kamenev, J. Smith, and S. Birchfield, "Toward Low-Flying Autonomous MAV Trail Navigation using Deep Neural Networks for Environmental Awareness," 2017.

[20] J. Collier and A. Ramirez-Serrano, "Environment classification for indoor/outdoor robotic mapping," *Proceedings of the 2009 Canadian Conference on Computer and Robot Vision, CRV 2009*, pp. 276–283, 2009.

[21] D. C. Asmar, J. S. Zelek, and S. M. Abdallah, "SmartSLAM : localization and mapping across multi-environments," *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, pp. 5240 – 5245, 2004.

[22] S. Saeedi, L. Nardi, E. Johns, B. Bodin, P. H. J. Kelly, and A. Davison, "Application-oriented Design Space Exploration for SLAM Algorithms," pp. 1–8.

[23] E. P. Simoncelli and B. A. Olshausen, "Natural Image Statistics and Neural Representation," *Annual Review Neuroscience*, 2001.

[24] W. S. Geisler, "Visual Perception and the Statistical Properties of Natural Scenes," 2008.

[25] C. Valgren and A. J. Lilienthal, "Sift, surf and seasons: Long-term outdoor localization using local features," in *3rd European conference on mobile robots, ECMR'07, September 19-21, Freiburg, Germany*, 2007, pp. 253–258.

[26] N. Yang, R. Wang, X. Gao, and D. Cremers, "Challenges in monocular visual odometry: Photometric calibration, motion bias, and rolling shutter effect," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2878–2885, 2018.

[27] P. Kim, B. Coltin, O. Alexandrov, and H. J. Kim, "Robust visual localization in changing lighting conditions," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 5447–5452.

[28] B. Risse, M. Mangan, W. Stürzl, and B. Webb, "Software to convert terrestrial lidar scans of natural environments into photorealistic meshes," *Environmental Modelling & Software*, vol. 99, pp. 88–100, 2018.

[29] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[30] J. Bruce, J. Wawerla, and R. Vaughan, "The SFU mountain dataset: Semi-structured woodland trails under changing environmental conditions," in *IEEE Int. Conf. on Robotics and Automation 2015, Workshop on Visual Place Recognition in Changing Environments*, 2015.

[31] J. Engel, V. Usenko, and D. Cremers, "A photometrically calibrated benchmark for monocular visual odometry," in *arXiv:1607.02555*, July 2016.

[32] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

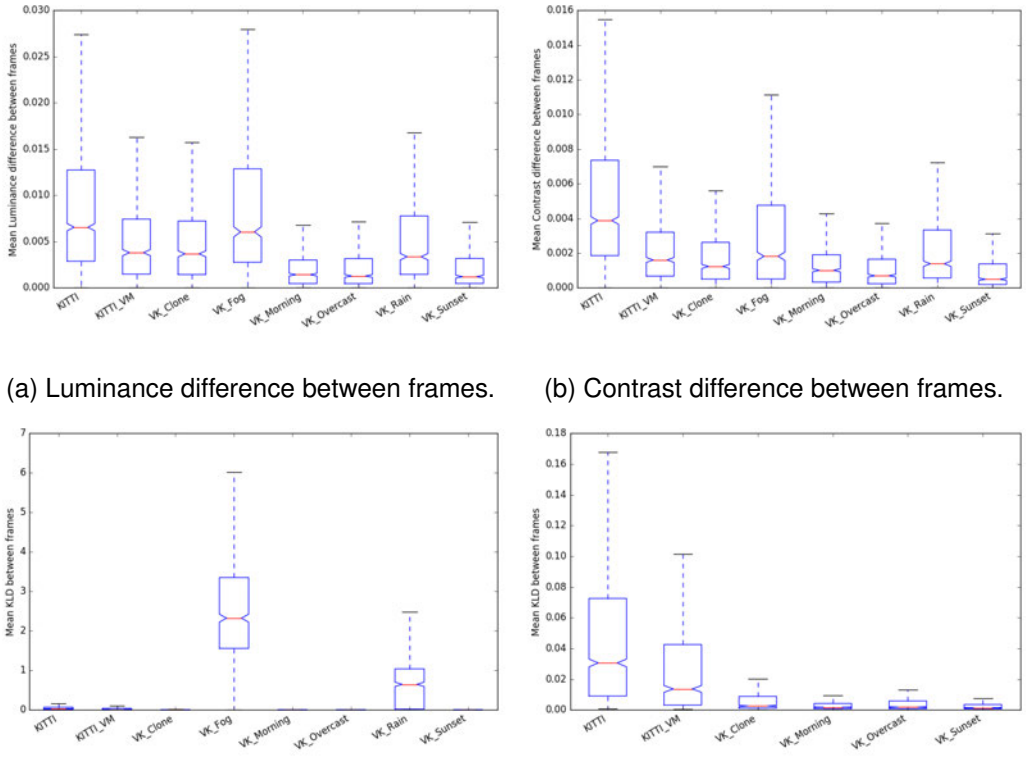
[33] M. U. GmbH. Broadleaf forest collection. [Online]. Available: https://www.youtube.com/watch?v=Zyq_UpOQ9r4

[34] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization," *International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[35] T. Stone, M. Mangan, P. Ardin, B. Webb *et al.*, "Sky segmentation with ultraviolet images can be used for navigation," in *Robotics: Science and Systems*. Robotics: Science and Systems, 2014.

4.4 Additional Experiments

Since submitting this publication, we became aware of Virtual KITTI (Gaidon et al., 2016), a synthetic version of the KITTI dataset. Videos from this dataset are intended as copies of the routes taken by the real vehicle, which allows us to run a previously unavailable comparison between real and simulated versions of the same tracks, further excluding the possibility that observed statistical differences are due to the motion of the camera. Virtual KITTI additionally includes versions of the same tracks under different simulated weather conditions.



(a) Luminance difference between frames.

(b) Contrast difference between frames.

(c) KL Divergence between frames.

(d) KL Divergence with extremes removed.

Figure 4.1: Scene statistics comparison between KITTI and Virtual KITTI.

Figure 4.1 shows a comparison of the scene statistics for the real world KITTI dataset with each of the virtual versions of the same data, including varying weather conditions. For luminance and contrast changes, we see the same effect observed when comparing real and synthetic forest data; the simulation appears very different to the

real video. Going back to check the virtual datasets, we found that they only recreate some of the KITTI videos, and so we added a comparison for a cut down version of real KITTI containing only those videos "KITTI VM" (for Virtual Match). The virtual data matches KITTI VM much more closely on luminance and contrast, but is still distinctly different on KLD.

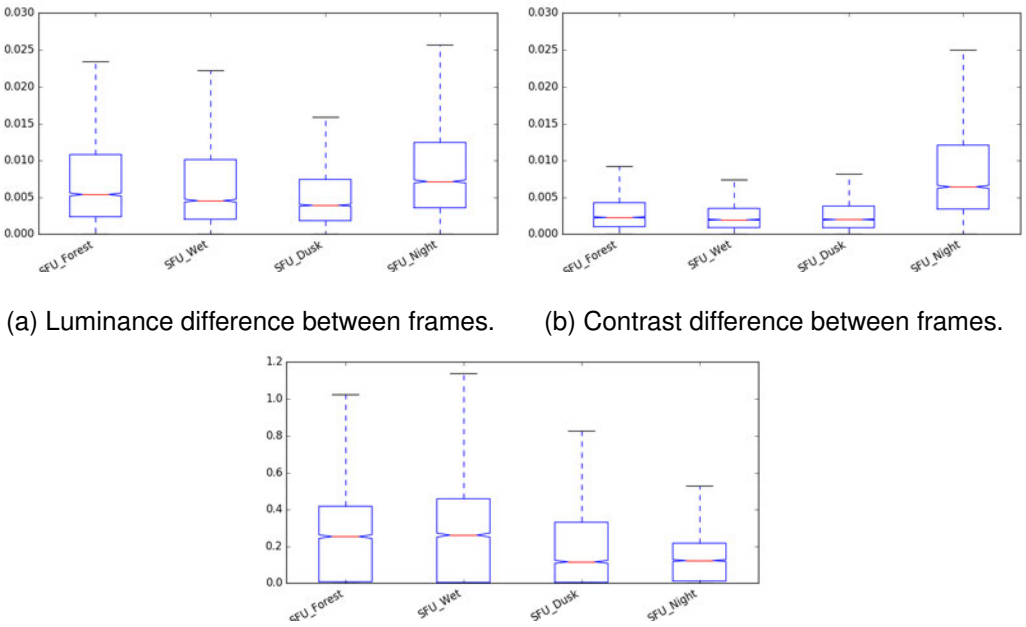
Between synthetic conditions, we see an expected difference reduction in luminance between the baseline "Clone" and the conditions with modified lighting ("Morning", "Overcast", "Sunset"). The stand out conditions are "Fog" and "Rain", which show notably higher values for all stats. The high KL divergence of these conditions is easily explained and expected, as they are much more dynamic scenes. The increase in lighting related stats is more puzzling, however, and we would not necessarily expect to see the same in real world fog or rain. It is unfortunate we do not have such datasets to compare against.

For comparison, we also calculated the scene statistics for the alternative versions of the SFU dataset not reported in the original paper. These too represent the same route under varying weather conditions, though these are not directly comparable to the conditions provided in Virtual KITTI.

As can be seen in Figure 4.2, there is much less variability in scene statistics between the different real world conditions. The most notable exception to this is the much increased contrast change measured between frames under night conditions. This is likely caused by the headlights of the robot amplifying the high contrast of foliage when they pass across it, but uniformly illuminating less complex textures like the dirt track the rest of the time.

4.5 Conclusions and Future Work

In this chapter we proposed a method, Visual Scene Statistics, for characterising the differences between the environments represented in datasets used for evaluating Robotic



(a) Luminance difference between frames.

(b) Contrast difference between frames.

(c) KL Divergence between frames.

Figure 4.2: Scene statistics comparison between SFU Forest in under different conditions.

Navigation. These statistics are measurable properties of images or videos which reflect conditions known to be challenging for SLAM systems; lighting, scene motion and heavy texture. They have been used by visual ecologists studying the adaptations of natural visual systems to different environments, but we are the first to explore their use in evaluating artificial vision systems, whose performance is also likely to be biased towards the kinds of environment in which they were primarily developed and tested. Our development of the libraries for calculating scene statistics over large video datasets is also of potential use back to visual ecology, where no such shared resource seems to exist.

We verify our visual scene statistics by using them to compare a variety of traditional datasets with those depicting forests. We also perform the first comparison of SLAM system performance in forested environments, where little previous navigation work has been done, and for these purposes we create new real and simulated forest datasets. Combining analysis of the environmental appearance with the performance

of the existing systems suggests a number of avenues for further investigation, outlined in the following sections. We also identify some limitations of the analysis so far, so we include further experiments to address these.

4.5.1 Initialisation and Feature Tracking

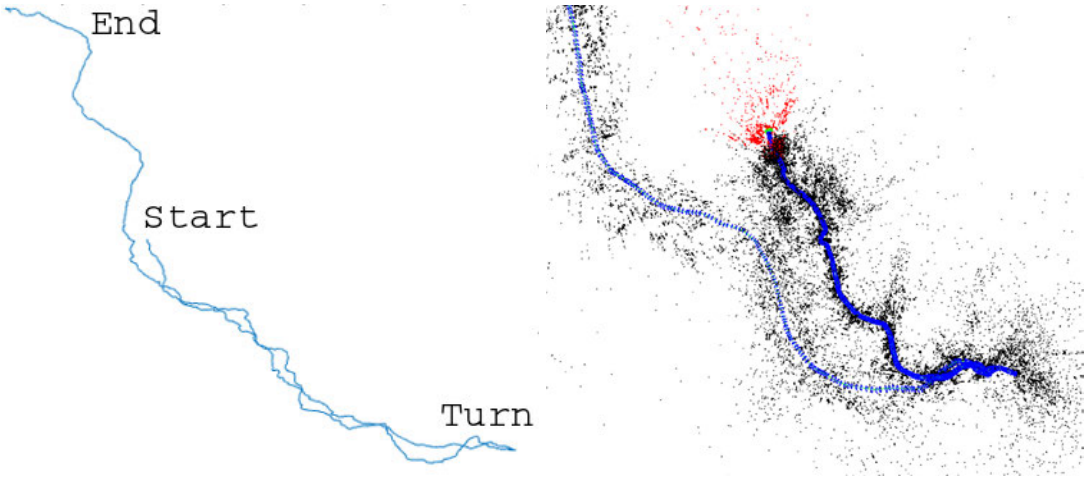
We have found in this work that SLAM performance, both direct and feature-based, suffers under forested conditions in terms of both initialisation and maintenance of frame to frame tracking. To follow on from this, we need to investigate further what is contributing to these failures, and whether these influences are indeed caused by traits of natural environments. This means a) providing favourable initialisation conditions to the systems, b) confirming that parameters are set appropriately, and c) seeking points of failure within the code. This is followed up on in Chapter 5.

Once we have a solid idea of the causes of poor performance in this environment, we can investigate pipeline changes or entirely different features which could perform better. As we have highlighted traits of forested data that are likely to be causing feature detection or matching to fail, we can be guided in what traits alternative methods must be robust against. We would expect performance improvements from features that are less influenced by lighting levels, or processing that leads to more consistent feature detection in a cluttered scene. This idea is explored in Chapter 7.

4.5.2 Loop Closure

We noted that despite much of the forest data revisiting locations multiple times, we observed no successful loop closures by the tested SLAM systems. Loop closure is essential for navigation and mapping performance at scale in the field, and can to some degree counteract inaccuracies in the tracking, so this is an important element to follow up on if we want to improve overall performance.

Convolutional Neural Networks (CNNs) such as NetVLAD (Arandjelovic et al.,



(a) GPS data provides rough ground truth but is inaccurate under canopy.

(b) Top down view of the map produced by ORBSLAM. The darker blue section is the return route.

Figure 4.3: Comparison between GPS ground truth and ORBSLAM map. Much of the structure of the trajectory is maintained, but the scale drifts making the longer return route appear shorter.

2016) have demonstrated promising performance on place recognition tasks similar to loop closure. We posit that with training on the right kind of data (contained plenty of vegetation) one of these CNN based approaches might be able to provide a more suitable representation than the hand crafted alternatives for place matching in forests. This idea is explored in Chapter 8.

4.5.3 Scene Statistics and Simulation

This work shows an initial correlation between the proposed Visual Scene Statistics and the "true" scene traits they represent, by analysing their outputs on environments we expect to demonstrate varying degrees of those traits. A valuable addition to this work, then, would be to run controlled experiments to more closely test the representative capacity, by setting up an environment in which only the trait is varying and tracking the statistic's change against it. Further use of a simulation is likely the only way to get this level of control.

The results presented in the paper, as well as in 4.4, lead us to be hesitant about the use of simulation as a platform for developing and evaluating improvements to navigation algorithms, especially when we want to be robust to real world conditions. Lighting, texture and motion all seem to stand out as significantly different in simulation, while performance of feature matching remains relatively good, implying that tests in simulated forests would fail to reflect the challenges of real forests.

Having established measures for key ways in which simulated data varies from real provides the opportunity to make targeted improvements to simulations and quickly assess how effective these have been by rerunning the tests. Incorporating a more complex sensor model with realistic noise might help with the relative predictability of frames, while lighting and scene dynamics are both also modelled aspects in game engines that can be altered. As our core interest does not lie in computer graphics, we leave these improvements open to future work by others.

Both the improvement of the simulation, and its use in verifying scene statistic representative power, are explored in Chapter 6.

Chapter 5

Investigating SLAM Performance in Forest Environments

5.1 Introduction

The work presented in Chapter 4 found that state of the art monocular SLAM systems struggled to initialise or track correctly on a variety of datasets from an environment not traditionally tested used for SLAM development and testing: forests. In this Chapter, we expand on our performance analysis by:

- Confirming best case system performance with:
  - Preferred initialisation conditions.
  - Tuning of relevant system parameters.
- Identifying the points in the data at which systems tend to fail, and the subsystems in the code which are responsible.
- Analysing how the scene statistics over the course of individual videos, rather than as summaries for whole datasets.

5.2 Optimising Performance

As a variety of factors could be affecting performance on prerecorded data, a thorough analysis of the capabilities of existing systems in a new environment requires that we attempt to control for the impact of variables not specific to that environment. In this section we first try running our SLAM systems with their preferred initialisation data to assess whether initialisation conditions in the previous datasets are limiting performance. We then use the available parameters on these systems to attempt to tune their performance further.

5.2.1 Ideal Initialisation

Existing forest datasets are not necessarily focused on providing an easy test case for SLAM systems, but rather something more typical of normal (unoptimised) usage. In particular, while initialisation conditions can be critical to establishing good baselines, the camera movements required to optimise these in the systems we tested are absent from the available datasets. For some systems, like DSO, initialisation failure is clearly an issue preventing the system from starting at all, but even in the cases where *some* initialisation is achieved, the quality of this can still be varied and have an impact on tracking performance throughout the video.

In order to assess to what extent initialisation was a factor in poor forest SLAM performance, new "ideal initialisation" data was collected. The new dataset, called "Forest Loop 2", was recorded using the same bebop platform, and in the same location, as Forest Loop. In addition, special attention was given to providing our tested SLAM systems with their preferred working conditions, which can found in the original papers (Mur-Artal and Tardós, 2017; Engel et al., 2014; Forster et al., 2014; Engel et al., 2017) and in the readme files provided by the authors' code repositories. For initialisation, this means:

- The starting location is a relatively flat and open area with no shrubs, to ensure

the scene is locally planar. This is an explicit preference of ORBSLAM, LSD-SLAM and SVO, although ORBSLAM has an alternative initialisation method running in parallel for robustness.

- The starting motion is predominantly translational, and in a series of circles perpendicular to the plane of the scene (in this case the floor). This is a stated preference of all four systems.

- The initial view is forward into the scene, rather than towards the ground or sky, as LSDSLAM specifies that the scene contain sufficient structure (multiple depths of intensity gradient).

As SVO's ideal conditions run counter to LSDSLAM's, and require it to be looking at the dominant plane of the scene (in our case, the ground), it has been excluded from further analysis here. Having a downward view of the scene, however successful the mapping might be, would mean losing the majority of the tree data that our forest monitoring use case would require. SVO's library requirements mean it cannot be compiled in the same environment as the other systems, so excluding it makes running tests much more streamlined. While gathering the data, LSDSLAM was run simultaneously in order to confirm successful initialisation and monitor performance. This was the system chosen as it has the most constraints, and other systems' requirements should be met by meeting LSDSLAM's. The same looped route as in Forest Loop was traversed, taking into consideration system preferences for, as identified in the Methodology:

- No purely rotational movements. This kind of degenerate motion doesn't allow new points to be triangulated, and so all turns should be accompanied by translation.

- No purely forward movements. Another degenerate motion, avoided by a combination of keeping the camera facing off at an angle from the direction of movement, and by strafing left and right along the route.

- No fast movements. All data recorded at slow human walking pace.

In total, 15 attempts were made at recording. 12 failed to initialise properly in the live LSDSLAM instance, and so the rest of the associated loop was not recorded. This left us with 3 completed loops (recordings 2, 6 and 7). Of these, recording 2 is the best lit, as the sun starting to set made the later videos darker despite there not being a noticeable change to the human eye. For this reason, we use video 2 as the ideal test case, but make the others available in the dataset for testing challenging lighting conditions in future work.

As the performance of most SLAM systems is not identical each time they are run on a particular input (discussed in Chapter 3.2.3) each system is run 5 times on the test loop video. We observe whether the system appears to initialise successfully, followed by how it performs mapping the rest of the route. The different systems have different performance profiles (for example whether they analyse the quality of their own initialisation) and so the qualitative criteria vary for each, as noted in the sections below. We are interested in whether, and if so when, they notably fail with the same parameters as Chapter 4 (default parameters, with number of features increased to 3000). Then, in Section 5.2.2 we will look at tuning these parameters to improve performance and finally in 5.3 we will look into failure modes in more detail.

A summary of results can be seen in Table 5.1. Observations on the performance of each system can be found in the following subsections.

Name	Successful Initialisation	Completed Track
ORBSLAM2	5	0
LSDSLAM	4	2
DSO	1	0

Table 5.1: For a total of 5 trials per SLAM system, the number of successful initialisations, and the number of times mapping continues to the end of the video.

5.2.1.1 ORBSLAM

ORBSLAM's failures are easily distinguished, as the system itself identifies when initialisation or tracking have failed and will then only resume if it can relocalise against the existing map. On the Forest Loop 2 video, initialisation was successful for all 5 trials, whilst the system never successfully tracked all the way to the end of the video. Tracking is always lost, twice at the first corner and three times at the second, and does not manage to recover. The second corner appears to coincide with a rolling shutter artefact in the image, ahead of which (see Figure 5.1) a tree trunk in the near field obscures some of the view around the corner to the right. A combination of these effects may be leading to ORBSLAM not having enough of the corner mapped to relocalise against it after the artefact causes it to lose tracking. This is in spite of the artefact being a small portion of the frame at any time, and disappearing before the turn begins. No artefact or similar obscuring trunk are present at the first corner, so while these may be factors in this particular failure the inability of the system to recover speaks to wider poor performance.

Figure 5.1: The approach to the corner where ORBSLAM fails, just before a rolling shutter artefact (Left), and the view after the turn that was partially obscured by a tree trunk (Right).

As seen in Figure 5.2, by the time the second corner is reached, the number of features being successfully tracked by the system is considerably lower than it is after a successful initialisation at the beginning of the loop (163 vs 559), and very few points are in the region in the top right that the camera will turn towards. This drop off is not

unique to this corner, and we see it to varying degrees on all our datasets. A small number of tracked points makes finding overlap in subsequent images, and therefore continued frame to frame tracking, more fragile. Eventually it becomes inevitable that a hurdle like an occlusion or sensor noise will temporarily reduce the number of helpful features further, at which point the system is not robust enough to recover.

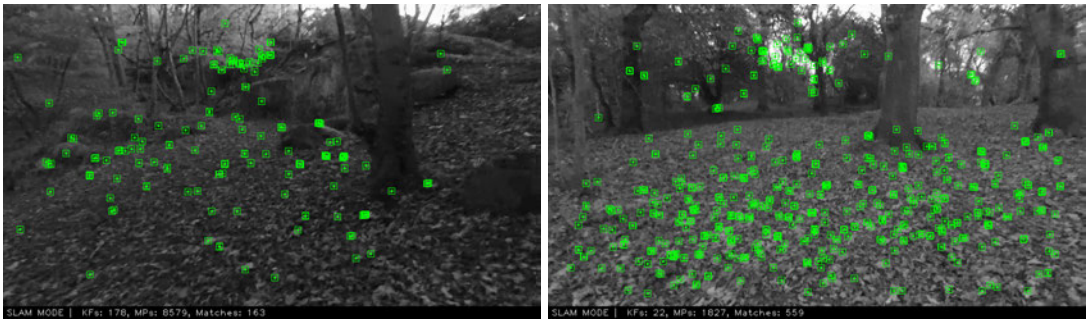


Figure 5.2: The features being tracked by ORBSLAM before the challenging corner (Left), compared to the number post initialisation (Right).

5.2.1.2 LSDSLAM

LSDSLAM's initialisation performance is not as easy to distinguish as ORBSLAM's, as the system does not identify its own poor initialisation. We instead judge success on whether the depth values for points in the image and point cloud seem to generally match the overall shape of the true 3D scene. This test failed in a single trial run (see Figure 5.3), with no apparent structure to the arrangement of points.

Route performance is also challenging to evaluate, as while LSDSLAM will identify lost tracking and try to relocalise, it does so quietly and frequently as part of even apparently successful runs, making the cases where it is genuinely struggling less apparent. Instead, we focus at how well the map recreates the shape of the route. Two of the tests succeed at mapping the full loop, although scale drift and no loop closure mean that the ends don't match up and the shape of the track only approximately matches the true path. The remaining tests fail at different points along the 3rd leg of the loop, as the relocalisation system incorrectly places the camera back on the initial

leg. This results in only half a map (see Figure 5.4). Images of the two locations seemingly being confused for one another can be seen in Figure 5.5. They do share some similarity, as both have the same approximate arrangement of two trees with respect to an otherwise open area. This loss of tracking and subsequent incorrect relocalisation seems to be the major issue for LSDSLAM much more so than initialisation, which is perhaps unsurprising given that it was the system used to confirm initialisation motions when the data was recorded.

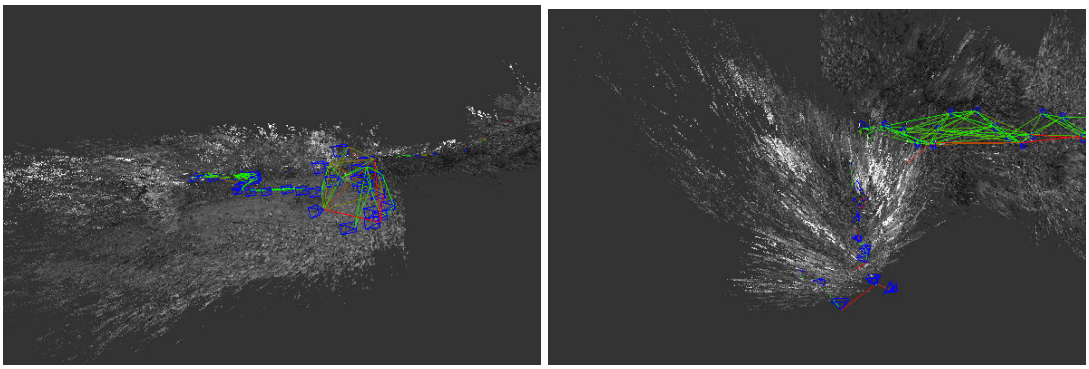


Figure 5.3: Examples of LSDSLAM successfully producing a reasonable initial depth map of the complex scene (Left) and the much less structured map produced when it fails (Right).

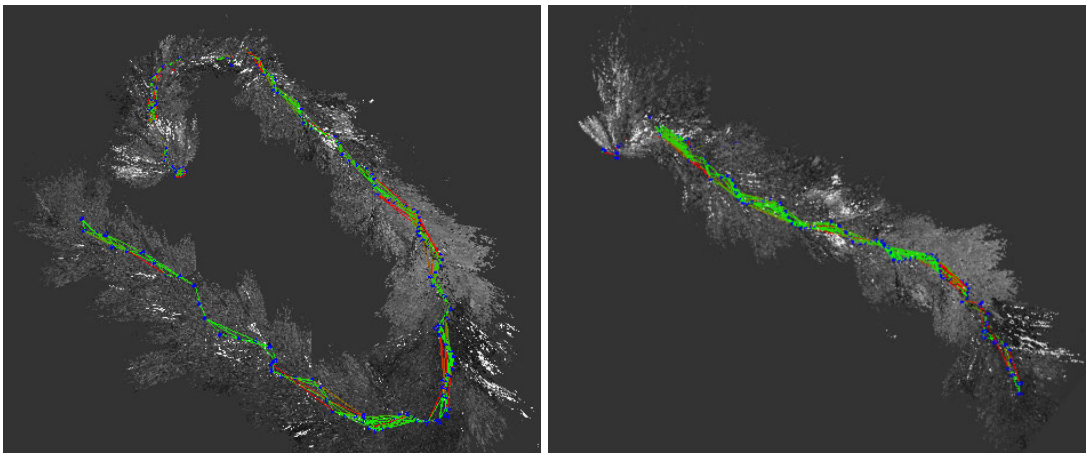


Figure 5.4: LSDSLAM: A successful loop with no loop closure (Left) and an unsuccessful half loop (Right).

Figure 5.5: The start of the Forest Loop 2 ideal video (Left), and the section towards the end of the 3rd leg that was repeatedly confused for it by the relocalisation system (Right).

5.2.1.3 DSO

DSO generally presents the same challenge for evaluating initialisation as LSDSLAM, in that the system can keep running even with a poor initialisation. In these tests, however, failures are still clear because in 4 out of 5 trials DSO crashed entirely while trying to initialise.

In the one test where initialisation completed, it did so obviously incorrectly, as can be seen in the maps shown in Figure 5.6. The points are spread out evenly across from each of the camera views, with no recognisable structure. The points being planar is reasonable for those on the ground, but the points of each of the two initial key frames shown here are projected above one another and at different scales. After this point, the system lost tracking and did not run any further. If we view the process from the image frame, as in Figure 5.7, we can confirm that the features extracted are indeed predominantly from the ground plane, and we can also note that the depth values (represented by the colours) are being incorrectly estimated. This is resulting in an estimate of the ground plane moving away from the bottom right hand corner of the image, rather than receding directly away from the camera. Initialisation is clearly a challenge for DSO, to the extent that we are unable to confirm at this stage how well it performs on subsequent tracking.

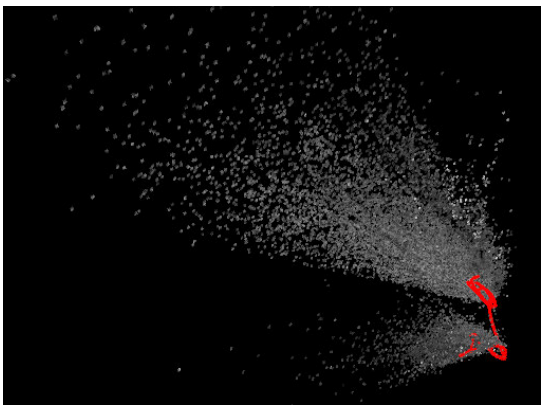


Figure 5.6: In the case DSO didn't crash entirely, failures to initialise correctly result in an unusable map.

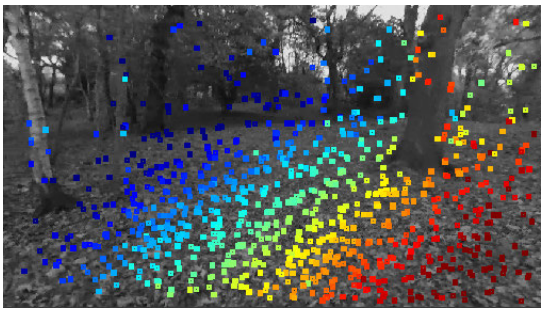


Figure 5.7: Depth map presented by DSO mid way through the initialisation movements. Note that colours represent depth values, and that the system incorrectly shows the scene as receding away and to the left.

5.2.2 Parameter Tuning

The default parameterisations for state of the art SLAM systems are likely to be biased towards the normal operating environments of those systems, and so giving these systems the best chance to perform in a new environment means attempting to tune them. An outline of the available parameters for each SLAM system can be found in 3.2. These experiments also allow us to reflect further on what the performance changes achieved by each parameter tuning might suggest about the particular challenges presented by forest data. This reflection is by necessity focused on changes to initialisation performance, as any parameter that results in poor initialisation cannot be assessed on the rest of the map.

5.2.2.1 ORBSLAM

Features: As covered in Chapter 4, we found in initial experimentation with ORB-SLAM that increasing the nFeatures parameter (controlling the number of features extracted from each image) beyond the recommended levels was necessary for achieving initialisation in complex natural scenes, and so have used a value of 3000 for the experiments presented both in that chapter and with the ideal data above. Having data that should now be easier to initialise on, we repeated 5 initialisation trials back at 2000 and found that in all trials the system is able to initialise.

Pyramid: ORBSLAM's feature extraction is performed on a pyramid of sequentially smaller versions of the input frame, to allow for detection of features at a range of scales. If the unreliable features dominating the forest scene are those on the corners of foliage, as is one possible cause of poor performance, then adjusting the scale pyramid to include less high resolution levels where these features are discernable would improve performance. To investigate this, we first tune down nFeatures to a level where the system does not reliably initialise, so that we can identify parameter changes which improve the performance enough to attain initialisations again. We choose the lower end of the recommended feature count, 1000, and confirm that 5 trials at this level all result in failed initialisation.

For our test configuration, we increase the scaleFactor parameter to 1.4. Compared to the original scaleFactor of 1.2, this will result in the first downsampled image in the pyramid (611 x 343) being very close in scale to the original second downsampled level (595 x 333), and the original first downsampled level being effectively skipped. We also change the nLevels parameter to 5 from 8, such that the smallest level of the pyramid (223 x 125) is close in scale to the original smallest level (240 x 134). This avoids any biasing of the results by inclusion of smaller downsampled levels than were originally present. 5 trials run with this configuration all result in successful initialisation, but all suffer from tracking failures at corners 1 and 2 similar to the base parameters. There are two possible explanations for the number of features being needed

for successful initialisation being lower with this reduced pyramid size. The first is that, as we predicted, larger scale jumps result in fewer high resolution images where the potentially unreliable foliage features are visible. The second is that, as features are extracted evenly across all levels and this pyramid change results in fewer levels, it allows for nFeatures to be lower while maintaining approximately the same number of features per level. This is worth exploring further, but in order to do so greater control over the input data would be preferable. As such, we continue in Chapter 6 after expanding our forest data simulation capabilities.

Corners: The remaining parameters for ORBSLAM, iniThFAST and minThFAST, which control corner detection thresholds, are recommended for tuning the system for low contrast environments. In the forest setting, where contrast levels vary more than traditional environments, the environment cannot be classified as purely low contrast. Adjusting these parameters either for high or low contrast could have a knock on effect on performance in the other case and lead to more failures. As contrast is a known issue, however, we try adjusting in both directions nonetheless. First we reduce minThFAST, which sets the threshold in the case that a first pass with iniThFAST did not result in enough edges, from 7 to 3. This will mean that if low contrast is causing few edges to be detected, the system will try again with a much lower threshold. The system fails to initialise in all of these cases, however. We then double both iniThFAST and minThFAST to 40 and 14, respectively. This sets the threshold requirements on the first and second pass higher, such that fewer low contrast corners will be used. Similarly though, this fails to initialise on all trials.

None of the above parameterisation changes are noted to have an impact on the tracking failure at corner 2 in the ideal initialisation video, so while ORBSLAM's parameters may assist in more reliable initialisation, reliable tracking and relocalisation at least remain open issues.

5.2.2.2 LSDSLAM

Noise: There are two parameters in LSDSLAM which we can adjust to try to account for noise, and particularly for this environment the effective "noise" present due to the large number of leaf edges. The first, minUseGrad, controls how strong an edge's gradient has to be for it to be included in tracking. We try increasing it from 5 to 10, which resulted in frequent tracking loss during initialisation and no initialisations we would consider successful. We also try reducing the value to 3, to see if instead the resulting increase in the number of edges provides useful additional information. This has a different negative impact on tracking, resulting in more frequent need to relocalise and more incorrect relocalisations. These usually only last for a single keyframe before correcting, but in one trial an incorrect relocalisation back to the start continued successfully, resulting in a second branch on the map. The other notable impact of minUseGrad reduction is more extreme scale drift, with a shrinking of each consecutive frame occurring from the start of camera movement after initialisation in two trials (example shown in Figure 5.8), and on the final leg of the route in another two trials. Overall, minUseGrad seems to be quite well tuned irrespective of the environment and considering more edges within this complex scene seems to produce bad depth estimates and tracking, while considering less makes initialisation difficult.

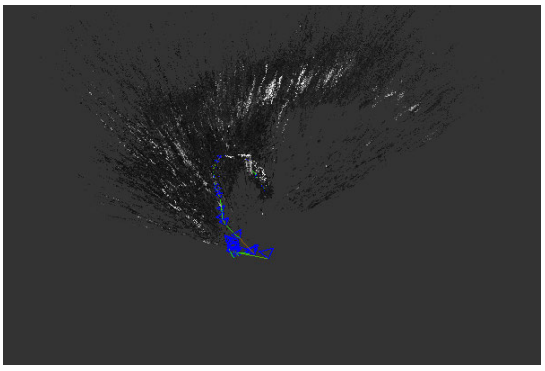


Figure 5.8: Example map produced by LSDSLAM with minUseGrad reduced, increasing the number of pixels considered for tracking. Four out of five trials displayed large amounts of scale drift and potentially poor depth estimates, as displayed here.

The second noise parameter, cameraPixelNoise, is designed to model sensor noise. Again, both increases and decreases could help here, as our camera may be more or less noisy than default, but also because it's possible there is overlap between the noisy environment and noisy sensor. We find doubling the parameter from 4 to 8 has no notable impact across 5 trials. Reducing it to 2 results in 3 trials with very notable drift (example in Figure 5.9), which seems to be caused by temporary losses in tracking, along with 1 trial that crashes during initialisation. The final trial seems to produce good depth estimates and map, at the higher end of the quality seen in the base conditions, but overall this change does not seem to be a net improvement. We conclude that this noise parameter is well tuned for our particular data.

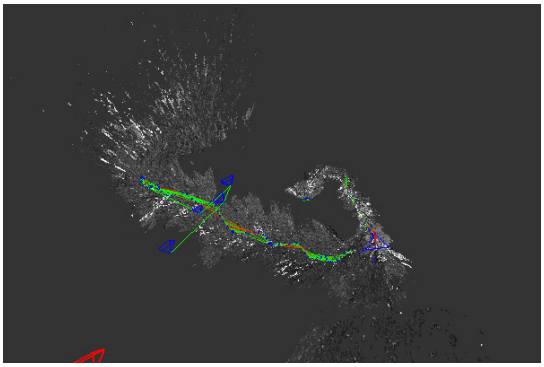


Figure 5.9: Example map produced by LSDSLAM with cameraPixelNoise reduced, reducing the expectation of noise from the sensor. Three out of four trials displayed very large scale drift, as pictured here.

Keyframes: Doubling both KFUsageWeight (which increases allowable overlap between keyframes) and KFDistWeight (which increases allowable distance between keyframes) increases the frequency and density of keyframes in our map. This pushed the performance of our test system to its limit, and indeed one of the trials crashed as a result. Two other trials saw incorrect relocalisations resetting the track to the start (seen in Figure 5.10), and the remaining two ran without notable differences to the baseline. It did appear as though less scale drift occurred over the course of the whole route than in the baseline case, but this is only a qualitative observation. It could simply

be an effect of more keyframes allowing for less drift to occur between keyframes, but this would still be a welcome improvement in case where the extra computing cost could be afforded. Similarly the increased frequency of keyframes could also be what is allowing incorrect relocalisations to more easily take hold and produce branches, where in baseline trials they seem to be corrected before frames are added. Alternatively, it could just be that more key frames in visually similar locations means more candidates for a mistaken relocalisation, and the reduced distance requirement skews this to the start of the loop where the initialisation is performed because the camera spends a lot of time there in a proximity which, under default parameters, would be too close for lots of keyframes.

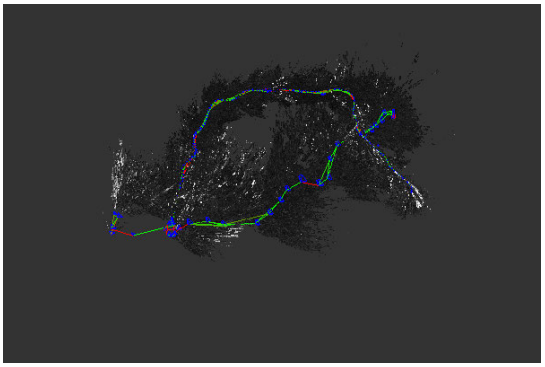


Figure 5.10: LSDSLAM with increased keyframe generation, resulting in incorrect relocalisation and continuation from the start of the loop half way through the track.

Relocalisation: The parameter controlling quality of matches required for relocalisation is relocalizationTH. We try both increasing this, to see if this avoids the frequent incorrect relocalisations we've seen in other tests, and decreasing it in case some previous poor tracking quality has been caused by failure to find any relocalisation where there is one that just has low match quality. Increasing to the maximum, 1, from 0.7 would require matches to be perfect for relocalisation to occur and so essentially result in the system stopping the first time it is required to relocalise, which 5 trials confirm. Instead we try an increase half way, to 0.85. The first of these trials produces good results and has no notable incorrect relocalisations. The remaining

four trials also seem to avoid incorrect relocalisation, but they also each have at least two instances of losing tracking and failing to relocalise where the base parameters had no such problem. LSDSLAM continues in these situations with an odometry-only link between the keyframes, and this causes large jumps in scale. We also lower the threshold to 0.4 from 0.7 and find no notable performance increases but all 5 trials have incorrect relocalisations. Three of these are in the usual location of aliasing on leg 3, although in one trial it occurs on leg 4 and on one it occurs multiple times in various locations (the branches created by this can be seen in Figure 5.11). Overall due to the large self-similarity of this environment, removing incorrect relocalisations likely requires tuning to a level where correct relocalisations are also missed.

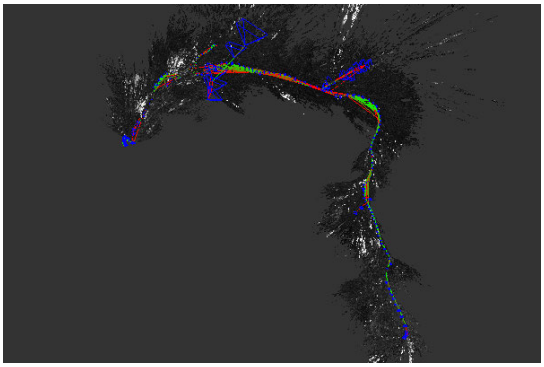


Figure 5.11: LSDSLAM with reduced threshold for relocalisation, resulting in multiple incorrect branches added to the track.

Depth Estimation: Disabling allowNegativedepths is recommended for a more aggressive culling of likely-incorrect depth estimates behind the image plane, at the expensive of also culling more correct estimates in the near field. As our video, especially during initialisation, does not have any features closer than 2 metres from the camera, this would seem like a straightforward improvement for us. Out of 5 trials disabling this parameter, 2 displayed initial depth mappings of good quality, with a narrower depth range and notably less near (red) depth estimates. These led into tracking of the same quality as previous experiments. The other 3 trials, however, failed to initialise properly, producing a mess of estimates, as can be seen in Figure 5.12. This

unreliability outweighs any potential performance increases, but it also raises a question about how a feature that culls near points would make a difference to performances on video with no near objects. This is perhaps a clue that, even in the cases where it does initialise, LSDSLAM is not correctly estimating depth for our forest data.

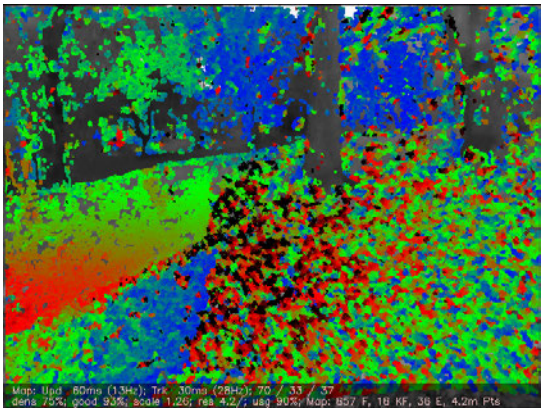


Figure 5.12: LSDSLAM with negative depth estimates (for points close to the camera) disabled, producing a poor quality depth map.

5.2.2.3 DSO

DSO is the most challenging system to perform tuning for, as the starting performance is the worst of the three. Given that in most Ideal Initialisation cases DSO does not initialise at all, and in the remaining one the initialisation is incorrect, we would not expect any small improvements gained by parameter tuning to be noticeable, as they would have to make a large enough difference that they allowed initialisation to complete. Complicating this further is that DSO's usual failure case is to crash, meaning that the video and map windows are not available to potentially provide additional information about how well initialisation was going at the time of failure. The log window does remain, and this is helpful in the later section where we look at failure modes.

Features: Similarly to the two previous systems, our first approach is to increase the number of features being extracted from each frame. In the case of DSO, this is

controlled by pointCandidates and paired with activePoints, which controls the total number of points being maintained. As before, we try doubling both of these, to 2000 and 2400 respectively. One of the trials crashes before getting any successful initialisation, and a further three briefly initialise before failing, still during the initialisation movements. One trial, however, manages to initialise and run long enough for the camera to start moving, before crashing part way along the first leg of the route. The initialised map in this case consisted of two unconnected ground planes, implying that there were still significant issues drawing matches between some of the frames. This is very similar to the single incorrect initialisation seen in the ideal conditions (Figure 5.6), and so while making a small amount of progress along leg 1 before failing does constitute an improvement over the default parameters, it is not clear the features are making a significant difference.

Keyframes: The parameters controlling keyframes in DSO are maxFrames, setting the maximum total, and kfFrequency, which controls how often new keyframes are added. We also try doubling both of these (though the maximum value for maxFrames is 10, so we're limited to that). Three trials crash during initialisation, as before, but this time 2 trials manage to initialise and reach the same point on leg 1 before failing. Looking further at this point in the video, it has a small rolling shutter artefact similar to the one at corner 2 that we hypothesise is causing ORBSLAM's failures. Similarly to the case for ORBSLAM, this suggests a poor robustness from DSO where it is unable to recover from temporarily losing tracking. Differently to ORBSLAM, the failure is catastrophic, given that it causes the same crash as other failures, but this leaves open the possibility that DSO could relocalise successfully if only it could be made to keep running.

Edge Gradient: The final potentially impactful parameter available to us in DSO is minGradAdd which, similar to minUseGrad in LSDSLAM, controls how steep the intensity gradient needs to be at a point for it to be considered for inclusion in the map. As before, this could potentially improve performance both by being increased (and

therefore only consider the strongest edges) or decreased (and therefore considering more edges). We attempt reduction from 7 to 4 first. Two of the trials crash during initialisation, one reaches the artefact on leg 1 before crashing, but the remaining two show results we haven't seen yet. One initialises seemingly successfully, but then immediately loses tracking, resulting in the virtual camera location spinning gradually into the distance in the map. This is assumed to be a bug in the map display, because the system should not actually be assuming continual movement when there is no tracking data. The final trial is the most successful seen so far; it initialises and manages to pass the artefact, before failing at the first corner. This parameter reduction does seem to be having a small improved performance on likelihood of initialisation, but the subsequent tracking performance is still very poor. Increasing minGradAdd to 15 results in similar performance as seen previously: 3 failures during initialisation, 1 just as it starts moving, then a final failure at the leg 1 artefact. There is no observed reason to use this increased threshold.

5.3 Failure Modes

In this section we attempt to trace the locations, in code, where the main failure cases identified in the previous sections seem to occur.

The evaluation of SLAM systems is typically achieved by demonstrating reduced error against a ground truth on benchmark datasets or possibly one newly recorded by the developers of the system. It is much less common that systems are evaluated by evidencing how subsystem performance varies with the presented input data, in favour of evaluating whole system performance at scale even when environmental conditions are being considered, as in Olid et al. (2018). Perhaps as a result of this system level focus in the field, combined with the time commitment required to do otherwise, research code is often sparsely commented, poorly documented, and generally not necessarily designed to be understood by a wider audience. Issue tracking functionality on GitHub

can help answer major questions, but this too is reliant on the continued commitment of the authors or other volunteers past publication.

Our approach focuses on obvious, "catastrophic" failures. The workings of SLAM systems are very complex, and the identification of small errors accruing over time, or errors separated in time from the failure they cause (e.g. mismatches between feature extraction at the start and end of a loop) is a great challenge. This is a limitation of the approach here, as these individually small but widespread issues (such as more frequent confusion between features to be matched, or small differences between their locations in subsequent frames) are very likely to be a part of the challenge of forest environments.

5.3.1 ORBSLAM

The previous sections have identified two types of failure for ORBSLAM. The first is during initialisation, and while its risk can generally be reliably reduced by increasing the number of features being extracted we might still glean some benefit from seeing why it occurs in the case of recommended feature numbers. To reliably recreate initialisation failures, we set nFeatures back to 1000.

Different trials have present two different error messages at initialisation failure. The first, "Wrong initialization, reseting" The second error message, "Track lost soon after initialization, reseting". The latter indicates that a successful initialisation (requiring two keyframes to be matched and a baseline established) was achieved, but then tracking was before at least 5 keyframes had been added to the map. This hard-coded threshold means that the system could technically be made to initialise more often by increasing the rate of keyframe addition such that 5 are reached faster, but would then likely just fail at a slightly later point. A "wrong" initialisation is either the result of the median scene depth being negative or, as in our case, too few map points (less than 100) in the map. So, while initialisation is managing to find correspondence between frames in these cases, the number of successfully tracked and matched points in that

map is too small to be reliable.

ORBSLAM's other common failure case, when initialisations are successful, is at corners. In particular corner 2 is the most frequent failure location, and though the rolling shutter artefact that occurs near it is very likely a factor in the failures, we also see similar loss of tracking at corner 1 where there is no such artefact. Watching the features in the video feed for the cases, we identify that there are far fewer features being tracked before failures occurred than there were during initialisation (as identified in Figure 5.2). At corner 2 the presence of a tree trunk in the near field also occludes some of the track in the direction of travel. It is very likely that the overall cause of the failure is a combination of these factors.

Tracking failures report a "TRACKING LOST" message, which reflects that change of an internal state variable to "LOST", the state identifying in ORBSLAM's core logic that the system should be attempting to relocalise rather than initialising or tracking. This state can be set both by failure to track or to relocalise, but in the first instance it must be set by a failure to track as otherwise no relocalisation would have been attempted. This in turn means that the failure occurred in the TrackReferenceKeyFrame function, which attempts to match the current frame against the current keyframe in the case that there is no valid velocity estimate or where projecting with that estimate does not result in a match. In all cases, this failure implies that the system cannot find significant feature matches from the current frame.

5.3.2 LSDSLAM

LSDSLAM's failures occur during tracking whereas initialisation under default parameters is reliable, likely due to LSDSLAM being the system used during recording to confirm ideal initialisation data. The main failure case seen during tests was an incorrect relocalisation, mostly occurring midway along leg 3. In a trial to investigate this further, we observe that the system begins to report "TRACKING LOST" at frame 4383, with the system message also mentioning less than 4% good points and "NOT

DIVERGED". This message is generated by the main trackFrame function in SlamSystem, responsible for the core tracking loop. A good point here is one where the residual between it and the existing point it is being matched to is small enough, so this message is confirming that the reason LSDSLAM starts to attempt to relocalise (and therefore does so against incorrect locations) is because the similarity match with the actual currently location is poor. Conversely, the match at the location being confused for the current one must have a sufficient number of good matches. For this confusion to occur, we would expect these locations to look similar to one another, which we explore in the next section.

5.3.3 DSO

Similar to ORBSLAM, DSO has demonstrated failures both at initialisation and during tracking, with the initialisation failures being the most frequent. It also fails catastrophically in both cases, making identifying the timing of an issue easier, but also disallowing any review of the video or map data at the time of failure. The system logging outputs are extensive, though most are not specific to the failure, so sifting through them is the best way to find the sources of problems.

The "Initial Error" message is the immediately relevant one, in the trials where failure occurred during initialisation. Crashes in these cases are also accompanied by an error indicating that an assertion checking for whether a "step" variable has become infinite. The failure of this assertion appears to be causing the crash, which seems like a failure of good software design if this assertion is being made without thought for catching the case where it fails. The probable cause for this is that the "HdiF" variable, which is one of the factors of "step" has become infinite. Documentation is non existent here, but this seems to indicate an infinite boundary on the Hessian, caused by poor depth estimates.

The failures on tracking occur very soon after initialisation, somewhere on the first leg of the route. Before failing completely, the system state displays "RE-TRACK

ATTEMPT" warnings, indicating that tracking has been lost in the main tracking function trackNewCoarse. This triggers attempts to track at progressively coarser levels until a match with a threshold residual is found or residuals start getting worse, and it seems to be during this that the failure occurs. System messages mentioning "REPEAT LEVEL" and "INCREASING cutoff to X" are also abundant here, but these indicate progression through progressively more coarse pyramid levels, and would be present to some degree in successful trials too.

5.4 Scene Stats Over Time

As a final method to examine our dataset in more detail, we can look at the scene statistics plotted for all frames rather than as a whole video summary. In this way, we can attempt to identify if particular extremes of the measured traits are present at the main points of failure, as opposed to compound effects over an extended period, or causes not measured by the scene statistics at all. The plots of scene statistics over time can be seen in Figure 5.13. Frame values presented as reference here refer to the frame numbers in the plotted data, not the real files (from which presented data has been downsampled for frame rate normalisation). We have removed the data from frame 2800 onwards, as these are large spikes from the mav landing and being shut down, not a part of the route. These do at least show, however, that no matter the environment some actions (such as the rapid motions or placing the robot facing a rock) can lead to very different readings from the scene statistics. The initialisation process ends between 560 and 700 frames. The challenging corner 2 is between 1240 and 1550, and the third leg runs from around 1500 to 2200.

Luminance changes appear to be larger during the initialisation section, which would fit well with this being in a more open area. The rest of the video seems to have the same general variability. Peaks in constrast are more spread out, making them easier to analyse. The first spike, around frame 240 (which is also a high point for

Luminance) is at the end of the initialisation as the camera tilts slightly upwards, and so seems to be caused by more of the sky entering view. The second, around 1200 and preceded by some smaller spikes, is the end of corner 2 and so coincides both with the artefact but also with the turn. The final spike (that isn't due to landing) around 2650 had no notable cause for contrast change when we reviewed the data.

KL Divergence is strongest during the initialisation period, perhaps due to the increased "surprise" or strafing sideways, where new parts of the scene enter the image, as opposed to largely forward motions where existing parts of the scene steadily grow in apparent size. There are also a few noticeable troughs in the KLD values, one potentially at $\tilde{1}300$ at corner 2, and the largest one at 1880 near to the frequent relocation aliasing. The video frame at 1880 is shown in Figure 5.14 compared against one from the start of the route, and some similarity is apparent in the structure. If drops in KLD are indeed preceeding both locations where tracking failure have most often occured, this would be a notable finding and potentially useful as a live predictive tool. It is not immediately clear why low KLD would preceed either of these failure points, but it is possible that the large amount of local self-similarity represented by the low KLD value is indicative of locations where feature matchers will find it particularly challenging to extract and match features accurately.

5.5 Conclusions and Future Work

In this chapter we extended our performance analysis from Chapter 4, where we had identified initialisation and feature matching as two potential problem areas (consistent with other works such as Yang et al. (2018)), and aimed to further narrow down the limitations of existing SLAM systems on forest environments. To support this work, we recorded a new Forest Loop 2 dataset, presenting ideal initialisation conditions not included in existing datasets like SFU Mountain (Bruce et al., 2015) to disambiguate whether poor initialisation was the cause of the previously seen poor performance, or

whether some can still be attributed to the forest environment.

ORBSLAM and LSDSLAM reliably initialise on the new data, a notable improvement compared to existing datasets, implying that initialisation motions had been a part of the challenge previously, in agreement with the findings of similar work (Chahine and Pradalier, 2018). Once initialised both systems track for variable lengths of time, and while they fail in most cases to complete the route, they do so in different locations. ORBSLAM's tracking failures are focused around either of the first two corners, while LSDSLAM's mostly occur along a straight leg with visual similarity to the starting location (also consistent with Chahine and Pradalier (2018)).

Our second set of experiments looked at the impact of tuning parameters related to the challenges these systems were experiencing, to assess whether the default parameters used in the original experiments were simply not suited to this environment. Most parameter changes made no noticeable difference or worsened performance. Increasing the number of features being extracted and maintained does seem to reliably improve the initialisation performance of ORBSLAM, as does increasing the scale factor between image pyramid levels. Whether this latter effect is due to the scale factor itself or the resulting number of images is worth investigating further, but either way increasing feature numbers and pyramid scale factor both seem like reliable ways to improve initialisation performance for ORBSLAM, without any notable impact on subsequent tracking. Increasing features increases the rate of initialisation for DSO as well, although it typically failed immediately afterward so it is unclear whether this is a practical improvement. Changing the contrast threshold for which points are considered edges is a feature of the parameters for all three systems, though it has different implementations, and in texture heavy scenes this might seem like a way to filter out unreliable corners, or to make sure reliable points are still considered in all the noise. For ORBSLAM and LSDSLAM these values seems well tuned and changes in either direction have a negative effect or none, and while for DSO reducing the gradient requirement does still mostly result in failed initialisation it did also produce the most

successful trail in all of our tests, with DSO reaching the first corner before crashing. Overall these results suggest that increasing the number of features considered (either via feature parameters or edge parameters) does have a positive effect on initialisation at least in forest environments, likely due to the large number of features in these scenes. When chosing features from such a large collection, considering more edges in each frame increases the chances that a sufficient number of those edges overlap between frames. LSDSLAM's major failure mode, incorrect relocalisations, can be reduced by increasing the threshold required for matching two locations, but at the expense of being unable to relocalise more often, resulting in a scale drifted or unconnected map. Improving LSDSLAM's performance here would seem to require either a relocalisation algorithm with a lower false positive rate, or a tracking algorithm that did not need relocalisations so often. This first of these is potentially covered by the results in Chapter 8.

Overall, the fragility of these systems feels like a theme of our results, echoing contemporary work in Wang et al. (2020). The right initialisation movements, recorded with a target system running, get systems to initialise more often. Increasing the number of candidate features considered counteracts the increased number of edges in the scene. These are hopefully good pieces of knowledge to be armed with when tailoring the recording and processing of mapping data in a new environment, but even after this tailoring none of our systems can reliably fully map an "ideal" forest sequence, and so for them to be able to robustly map unknown forest sequences in real world scenarios undoubtedly further work is needed. This work needs to cover more reliable frame to frame tracking (Chapter 7), more accurate relocalisation (Chapter 8) and to make it as effective as possible we need a better understanding of the traits of forest data, along with the ability to generate it in a controlled manner (both in Chapter 6).

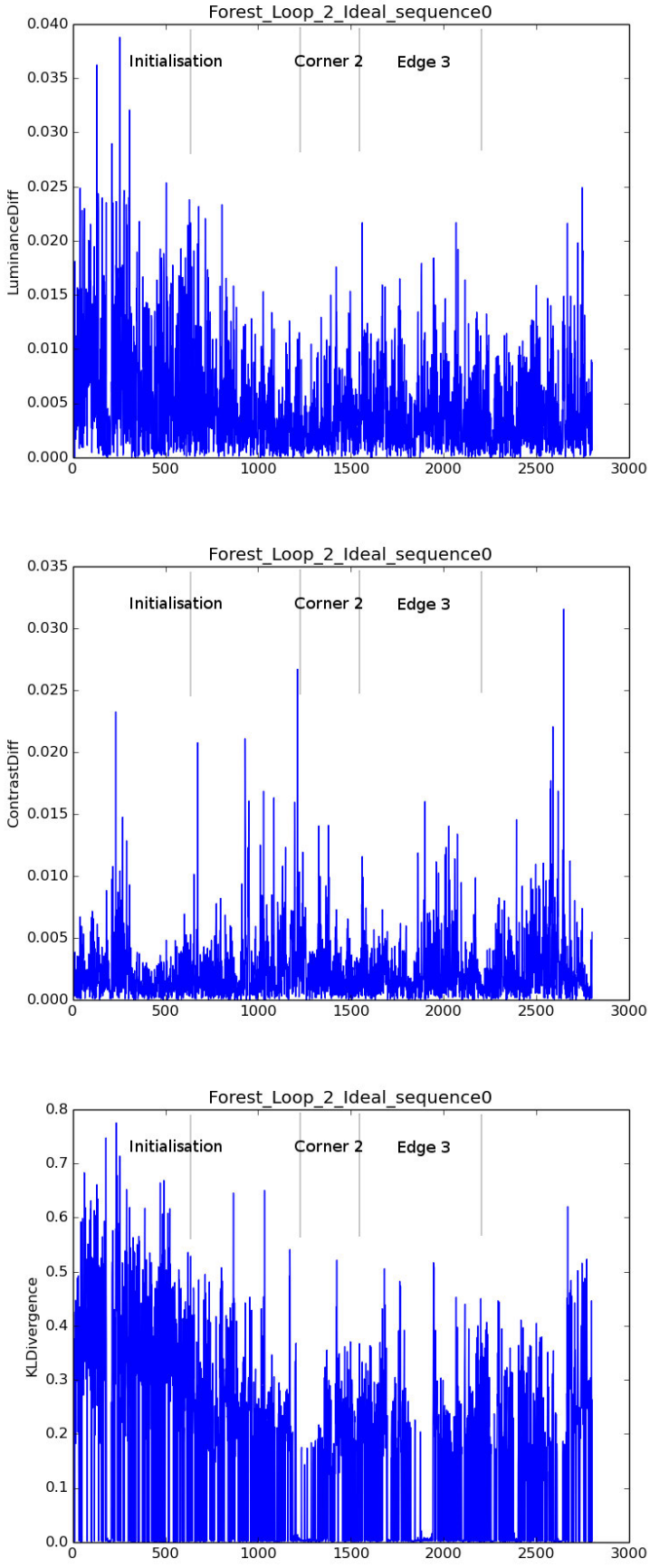


Figure 5.13: Scene statistics displayed by frame.

Figure 5.14: The corner at which a notable KLD trough occurs (Left) compared against a frame from the start of the route (Right).

Chapter 6

Reproducing Environmental Traits in Simulated Forest Environments

6.1 Introduction

Unlike real world forest recordings, simulation allows for testing the impact of varying environmental traits on otherwise controlled scene data. This allows us to investigate two connections: first, how well our scene statistics reflect the environmental traits they are intended to, and second, what the impact of individual environmental traits is on the performance challenges outlined in Chapter 5.

Given all of the challenges using real world data for navigation development (such as ground truthing, or safety to operators or platforms), it is unsurprising that a variety of simulation environments have been developed, such as AirSim (Shah et al., 2017) for roads, and ICL NUIM Handa et al., 2014 or Habitat (Szot et al., 2021) for indoor scenes. These simulators can potentially produce as much data as the user needs, with the integration of procedural generation algorithms, but as identified in Chapter 4, it cannot be assumed that simulated scenes will provide the same challenges as real visual data.

As noted in Chapter 2.1.4 approaches to addressing the sim2real gap vary, involv-

ing further processing of the simulated data once it is created, modification of the algorithms making use of it, or attempting to increase the similarity to the real world of the simulation itself. We take this last approach, and set out another goal for the simulation extension work presented here: to identify whether the simulation to real visual similarity gap is being closed by the improvement of the simulation.

In this chapter we:

- Improve our forest simulation, including the capacity to change environmental traits impacting appearance such as wind and snow.
- Evaluate the effect of simulated traits on the scene statistics, analysing where this might reveal limitations of the statistics or the simulation, including comparing the new simulation with the old in its ability to reflect real world data.
- Investigate the performance of SLAM systems across the new simulated conditions, focusing on initialisation as a repeatable test case.

6.2 Upgrading the Simulation

To produce a full simulation environment combined our generated forest with AirSim (Shah et al., 2017), a simulator designed for testing drones, cars, and other robotic applications in relatively large scale open environments. AirSim is also built using the Unreal Engine, making importing the forest generation package straightforward, and leaving open the option of quickly importing other forest packages by the same author (MAWI United(GmbH, 2022)) in future. The Unreal Engine, as well as the example forest layout provided by MAWI Unite had been upgraded since our first simulation. These updates include the various “winter” variables allowing for ground coverage snow and snowfall.

The integration of AirSim has a number of potential benefits for improving our simulation. First, AirSim has a richer simulation of physics, including for flight control

of virtual quadrotors, which we hoped would allow us to generate more natural camera motion through the scene. Secondly, AirSim provides support for simulated camera models (as well as other sensing modalities we aren't currently using), so we can match our virtual camera's lens properties to our real cameras. Thirdly, AirSim has a rich API allowing for communication between the simulator and any tests scripts we need to create. This allows for more fine-grained control over data generation and recording (such as "playback" of previously traversed routes and recording of video and ground truth data) that would be much more challenging to do within the simulation code itself.

Reliably being able to re-fly the same route through the environment while altering other conditions is a key feature for running some of our comparison experiments. This is achieved by two simple scripts, Record and Replay. The Record script logs the poses of the camera as it traverses the environment under manual control, allowing the experimenter to define a path once for reuse later. The Replay script can then run through the logged poses, sending movement requests for the simulator to repeat them. Alternatively to using Record, we can also define a shape by key way points such as corners, which is particularly useful in recording a route that returns to its starting location, as a human pilot can easily get lost.

We faced a challenge because neither of the two available API methods for visiting a camera location worked perfectly for our usage. The first allows us to set a pose directly, and have the camera jump to it, but even if we do this at the frame rate of the recording it still visually ends up jerky in motion. It seems as though this function is still impacted by the physics engine, and that the sudden movement produces momentum each time causing it to overshoot. The second method allows for setting a destination and having the simulation move there itself, but this method only takes a location as input and not the full pose, assuming all travel is done with the camera directly forward and not the holonomic movement of a UAV. It is possible to set camera offsets to avoid purely forward video, but this still doesn't allow for complex pose. In

the context of the work we're trying to do, this latter limitation has much less impact on the quality of the videos, and so we chose the second API function.

We also experienced unexplained errors with height control in playback situations, where sometimes the camera was climbing rather than keeping to the specified heights. We managed to work around these, but for further robustness the simulator would benefit from the implementation of active height control to be able to maintain constant height over complex terrain. One way to do this would be to integrate a full drone flight controller and height sensor, but doing this full live controlled simulation was too complex for our primary purpose of recording data.

6.2.1 Simulated Environmental Conditions

The combination of the Unreal forest environment and AirSim allow for the simulation of various environmental conditions related to the real forest scene traits we want to explore. This means the simulation gives us a unique opportunity to vary these conditions while controlling other variables, such as the structure of the scene and the route of the camera through it, which would be practically impossible to control in a real forest. The conditions we simulate, and the scene statistics we expect them to impact, are:

Wind: One method for increasing the amount of motion in the scene is to apply a stronger wind factor. Wind is controllable both in terms of wind intensity and rate of change. Neither of these variables has units, but initial tests found that a maximum of 10 for wind intensity and 5 for rate of change resulted in notable but not unrealistic or seemingly dangerous levels of wind. Simulated wind effects all foliage in the scenes, including some visible swaying of the tops of tree trunks at 10 intensity. It does not, however, impact the flight of the vehicle where in the real world this would be a confusing factor for the effect of wind on the data. We adopt a notation for wind of "X-Y" where X is intensity and Y is rate of change, for example "Cloud_5_10" is a video in cloud conditions with 5 wind intensity

and 10 wind rate. The default for either in the baseline simulation is 1, so we don't include the numbers if either value is 1. Where only one number is given, it is always intensity. Pictured in Figure 6.1b is wind value 10, and rate can't be seen in single images so this is not pictured.

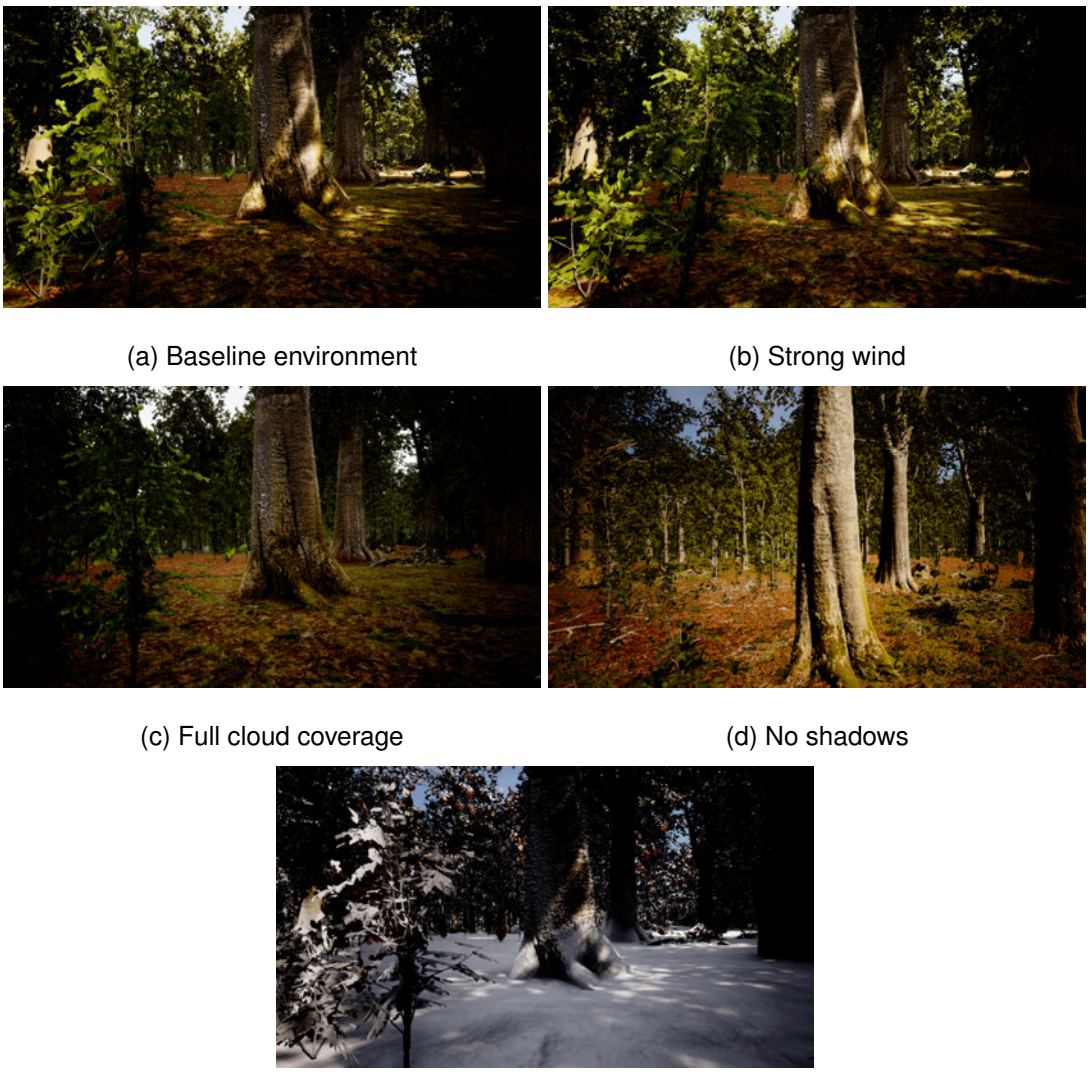
Clouds: To remove the inconsistent lighting effect of sunlight through canopy, we can modify cloud coverage to 100 percent. This diffuses the lighting from the simulated sky, and makes scene lighting consistent, unaffected by canopy cover, as seen in Figure 6.1c. A similar impact could be achieved by turning off the directional lighting from the simulated sun, but this would require a further step in reintroducing sufficient replacement lighting, so the cloud approach using a single variable was easier to implement.

Shadowless: Discovered later as a possible alternative to Cloud, the engine allows us to turn off the shadows caused by the sky light. This normalises the lighting considerably across the environment, so similarly to Cloud it has the effect of removing the inconsistent canopy lighting. As a down side, it is not limited to only the shadows caused by the canopy, and so results in a scene with a lot less shadows.

Season: To reduce the amount of complex texture in scenes, we can utilise snow depth to cover the ground, as seen in Figure 6.1e. It is also possible to change leaf colours to reds and browns in order to modify the colour space of the scene, though this function is not used in this thesis. An alternative approach to texture reduction considered was in removing the leaves from the trees. Unfortunately the way these models were constructed (from 3D environmental scans of real trees) means there is no in-data distinction between different elements of the trees aside from the trunks. This means producing leafless models would be an extensive manual task, requiring familiarity with graphics tools, and is beyond our scope here.

Wind, and Cloud are controlled through variables in the "MPC_BF_Controller" file, and the variables "MW_WindIntensity", "MW_WindSpeed", and "MW_CloudsIntensity". Snow is activated by "MW_Winter" (1 to activate) and "MW_SnowMultiplier" (which also changes leaf colour from green at 0 to red at 1), then the extents of the snow are controlled by "MW_SnowPower" (our Winter scenes all use 8, which we found to give full coverage).

We explored a second method for adding more motion to the scene; particle effects. Rain, falling snow, or swirling leaves are all available within the simulation with controls for the density of particles. A side effect of these is that they also add in additional texture, so they don't allow us to test motion independent of this. A more practical issue is that the design of AirSim ties these particle effects to a local area around a vehicle (presumably for processing efficiency), meaning that the freely controllable camera needed to recreate routes reliably for recording cannot benefit from having these effects follow it. For both of these reasons, we do not use these effects for now.



(a) Baseline environment

(b) Strong wind

(c) Full cloud coverage

(d) No shadows

(e) Snow covered ground

Figure 6.1: Our simulated forest environment allows for a variety of environmental conditions.

6.3 Comparing Scene Statistics and Environmental Traits

Our scene statistics, as presented in Chapter 4, are proposed as measures reflecting particular traits of natural scenes, based on the use of similar statistics in visual ecology literature (Geisler, 2008). Until this point we have been confirming how well the statistics are fulfilling this role primarily by how well their outputs match our perceptions of the variations of traits within video datasets. With the development of a more

feature-rich simulation, we have the opportunity to control particular traits of the environment while keeping others fixed in a way that is essentially impossible with real data. In this section, we use this new capability to further assess the usefulness of scene statistics in reflecting difficult environmental traits, by isolating environmental conditions we expect to influence single scene statistic and then running experiments varying that condition to confirm similarly varying measurements.

The environmental conditions and expected corresponding scene statistics investigated here are:

- Full **cloud** cover, which has the impact of reducing the direct lighting from the sun. In turn we expect this to reduce the impact of the varying patches of direct sunlight or shadow caused by light filtering through the moving patchwork of leaves in the canopy. We would expect this more consistent lighting of the scene to result in smaller **luminance and contrast changes**, as both luminance and contrast are derived from the brightness of the images (Geisler, 2008).

- We later added **shadowless** as an alternative approach to removing canopy shadow and hopefully generating the same conditions expected from cloud cover.

- Variable **wind**, controllable both in its intensity and rate of change, which impacts how frequently foliage changes its direction of movement and the extent to which it deforms in that direction. We would expect this increased in-scene movement to also increase the measured **KL divergence**, as an approximate measure of scene structure and motion (Saeedi et al., 2017).

- A **winter** mode where snow covers the ground layer of the scene, removing most of its complex leaf and grass texture. We would expect reduced texture to result in a reduced **variance of the Laplacian**, which is a measure of the strength of edges within the scene (Maini and Aggarwal, 2009).

The choice of environmental conditions is heavily influenced by functionality available within the existing simulation, and does not necessarily represent ideal isolation of

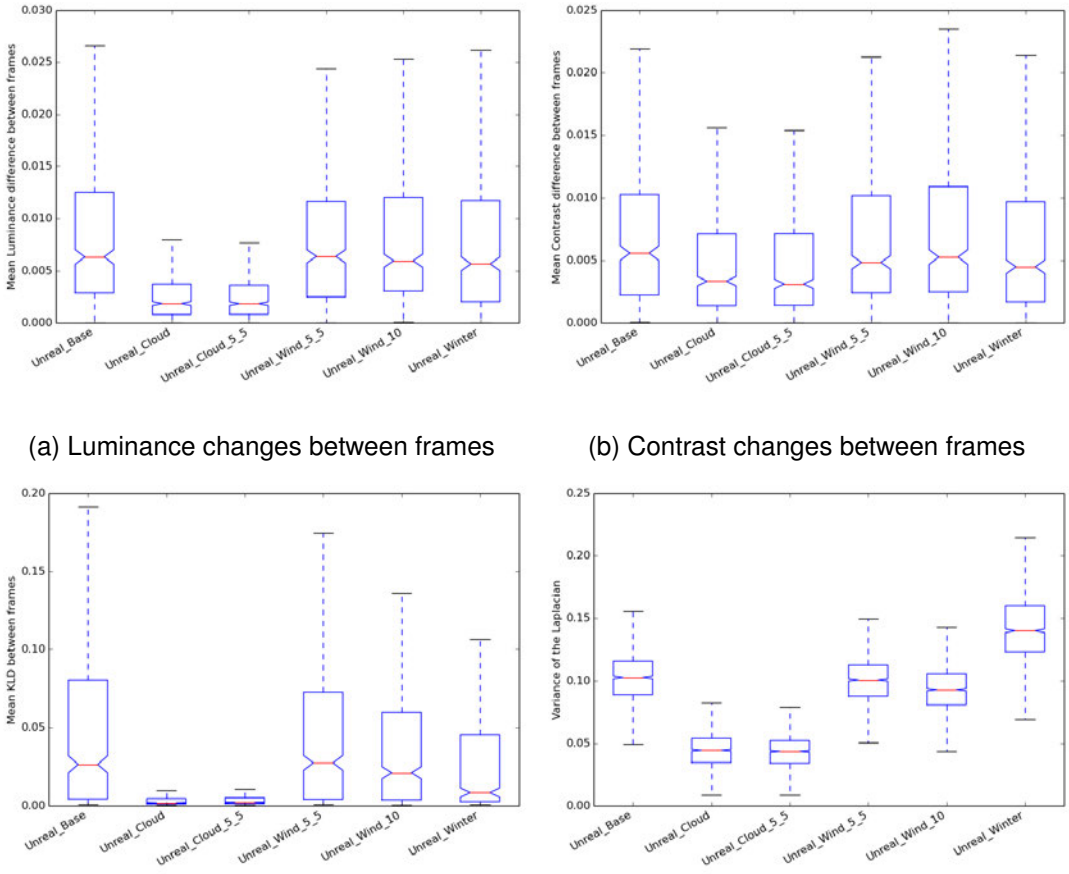
individual scene traits. Texture complexity, particularly, is difficult to isolate without fully re-rendering the simulated environments to have flat colour surfaces (and that's assuming we don't want to be able to vary it), and so the Winter condition is presenting a best approximation.

The route flown through the virtual environment is a squared loop intentionally similar in length to our real Forest Loop datasets. It is also traversed at a similar slow walking speed, 2mph, and similar height (1m). As the simulation handles camera height in world coordinates rather than distance from ground, it can get higher or lower when the ground height changes, varying by approximately -0.5 to +1.0. We start each recording with an initialisation procedure, flying from the rest position up to the 1m height before strafing right and left 5 times to give plenty of opportunity for SLAM systems to establish a baseline. The first leg of the square is repeated at the end in case the overlap proved useful in analysis of loop closures later, resulting in 4 turns and 5 straight legs total. Due to the stochastic nature of the simulation, the lengths of the videos still vary slightly, between 1449 and 1822 frames (around 1 minute). The difference in length from the Hillwood videos suggests the virtual agent is still moving faster than the real one, where the human experimenters' ability to traverse the difficult terrain is a limiting factor.

6.3.1 Scene Statistics of New Simulated Data

We generated scene statistics for the recorded routes through the simulated forests under each environmental condition, using the same EnvisionEd setup as we used in Chapter 4. These are presented in Figure 6.2 below.

As we can see in Figures 6.2a and 6.2b, the luminance and contrast changes between frames stay consistent for all environmental variations except for increased cloud cover, which matches our expectation that dappled light through the canopy contributes to large lighting changes. The impact on luminance changes are more pronounced, and we should be hesitant about drawing as strong a conclusion about



(a) Luminance changes between frames

(b) Contrast changes between frames

(c) KL Divergence between frames

(d) Variance of the Laplacian

Figure 6.2: Scene statistics for simulated datasets varying only environmental traits.

contrast here. Wind makes no significant difference to lighting on its own, nor does it impact the effect of Cloud when combined with it.

The results for KL Divergence show the opposite trend to what we would expect if it is acting as a measure of unpredictable motion in the scene. To the extent that wind is having any impact on KLD, it seems to be that higher wind speeds actually reduce KLD values, but the extent of this is not enough to be conclusive. The cloud conditions have a much more notable impact on KLD than any others, when we would expect them to have none. This implies that something about the measure is being influenced more heavily by the lighting of the scene than it is by the motion. As the primarily black and white Winter condition also displays some reduced KLD, a related explanation could be that both Cloud and Winter are reducing the palette of colours (or intensities)

represented in the scene such that there is less opportunity for the "surprise" measured by KLD. Alternatively, the movement of patches of light through the canopy might be responsible for the high KLD usually and the absence of these in Cloud conditions reduces the result, but this movement shouldn't be occurring in the Base condition where there is no wind moving the canopy.

Is camera motion affecting scene statistics?

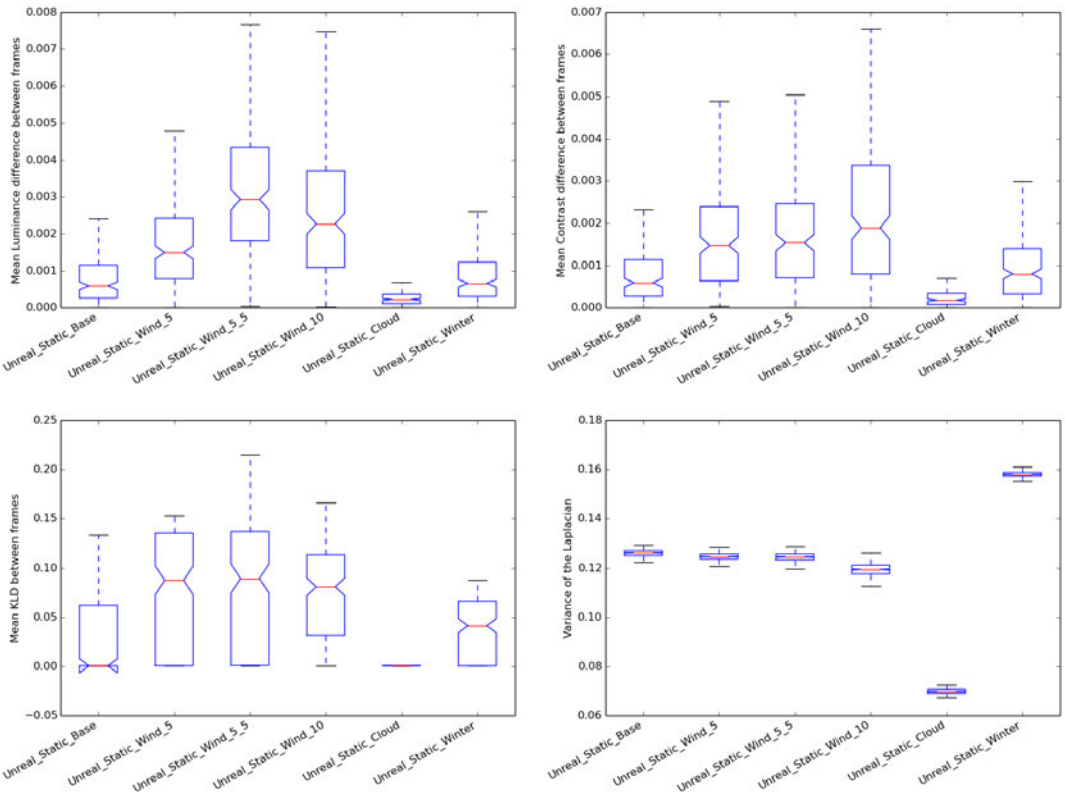


Figure 6.3: Scene statistics for versions of the Unreal Traits simulated conditions where the virtual camera holds a static position.

To further explore the impact of camera motion, we recorded alternative videos from each simulated condition with a camera hovering in place for 1 minute, and the scene statistics for these are shown in Figure 6.3. These show that Wind does have an impact on luminance, contrast and KLD, with the resulting in-scene motion moving the canopy such that direct lighting changes, as well as moving the surfaces that might

catch that lighting. Given that these effects are much less apparent in the moving data, it seems that the changing location or the movement of the camera could be having larger impacts and subsuming the effects of scene motion. Looking at the KLD values over time for the Base conditions confirms that the highest values typically come with the turning motion at corners. What this also reveals is that seemingly camera motion is not only adding to the KLD, but actually hiding the existence of greater in-scene motion in the Wind conditions.

Are changing colour palettes in Cloud and Winter conditions affecting scene statistics?

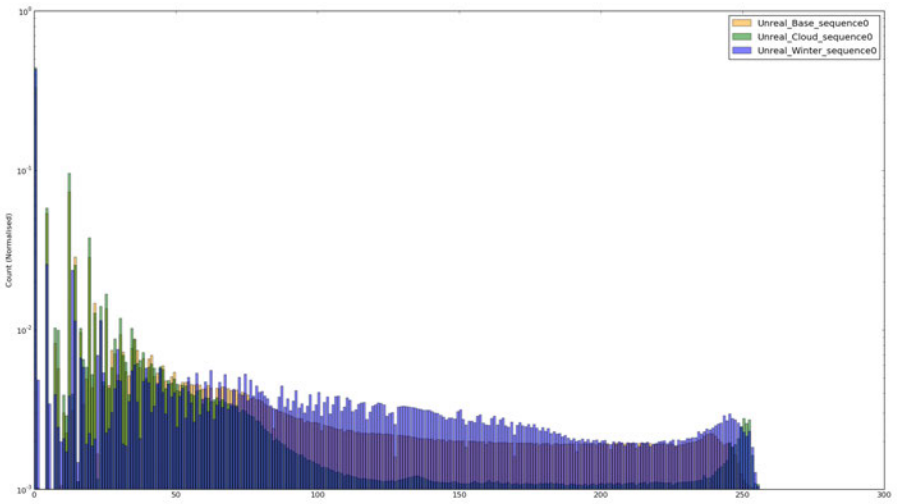


Figure 6.4: Histogram of all pixel intensity values for whole simulated video sequences, normalised so that heights sum to 1. Note the logarithmic scale.

To assess whether a constrained colour palette is influencing Cloud and Winter's low KLDs, we summed the intensity histograms KLD is calculated from and created a normalised histogram for each video. In Figure 6.4 we show the histogram for the two simulated conditions in question, alongside the baseline. Though all conditions have a large frequency of dark pixels, this is much more pronounced for Cloud, which

also has very few light pixels except for a spike for the sky and 44% of its pixels being pure black. This prevalence of dark pixels would make the intensity histogram of each frame similar to the last and therefor lead to low KLD. For Winter, the issue is much more likely to be specifically the colour palette being used, as the snow means that even though intensities are spread out over the full range these represent almost all shades of grey.

The results for the variance of the Laplacian in Figure 6.2d are also unexpected. Given that this measure is intended to capture the amount of texture in the scene, we expect that the removal of ground foliage texture in the Winter condition would result in a lower variance than other conditions, yet it is significantly higher. Unfortunately, the problem we are likely encountering here is that the statistic is not specifically measuring number of edges, but how varied the sharpness of the edges present is. In a scene with a lot of pixels close to both black or white, there will be a lot more maximally strong black-to-white edges, and so if the scene still contains a range of weak edges as well, the variation could be increased compared to the less well distributed intensities for Base and Cloud, as seen in Figure 6.4.

The variance of the Laplacian for the Cloud condition wasn't expected to be significantly different from base, but given that the patchy lighting through canopy is removed by clouds, it was plausible that any additional texture provided by these patches would be removed and the variance slightly lowered. This impact appears to have been stronger than expected, with the variance being significantly lower for Cloud than for the baseline condition. Similarly to Winter, this could be explained by the distribution of intensity values, and we can again see from Figure 6.4 that Cloud presents generally darker pixels, with 44% being pure black, compared to Base's 34%. Cloud also has more pixels over 246. In this case, an overall darker scene with only a few strong black-to-white edges and dominance of weak dark-to-dark could explain a smaller variance of the Laplacian.

Looking at the Laplacian edge images produced on early frames of each of the

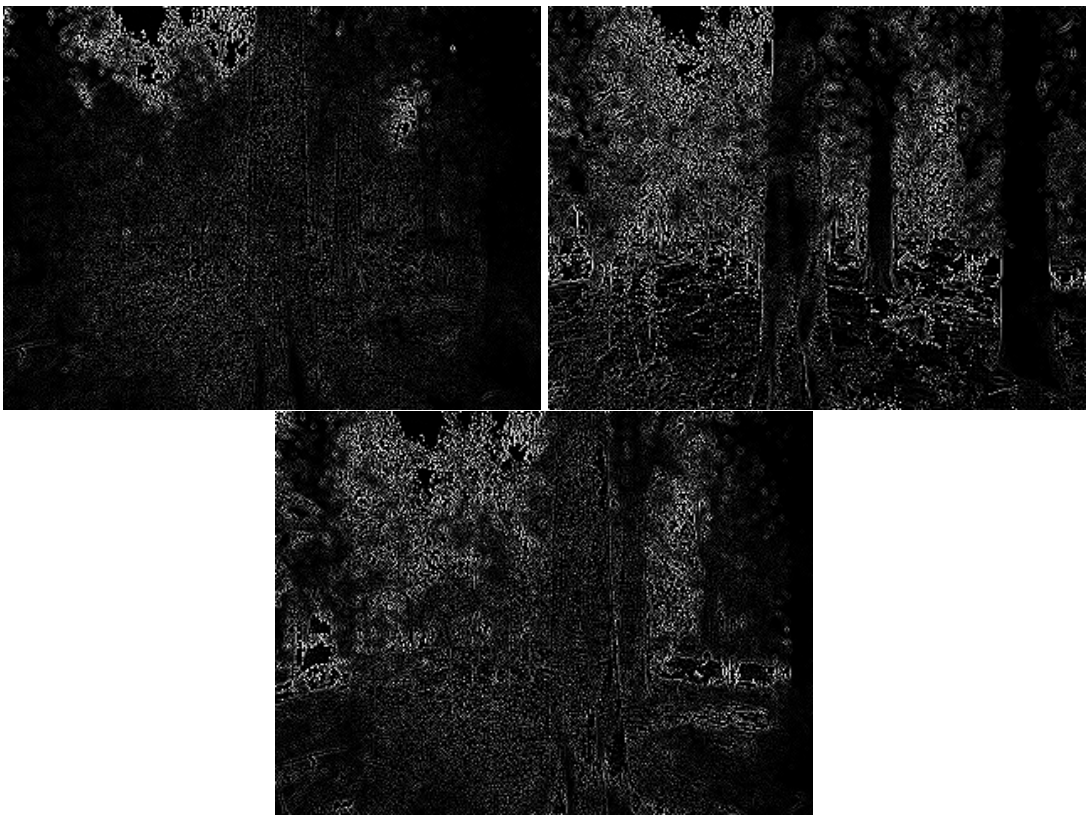


Figure 6.5: Laplacian edge images from the start of simulated sequences under different conditions: Cloud (top left), Winter (top right) and the base condition (bottom). Whiter pixels represent stronger edge values.

environmental conditions (shown in Figure 6.5) seems to support our initial interpretations. The Cloud image has notably fewer strong edge returns than the base condition, and they are limited to the top part of the image, where the leaf edges are contrasted against the sky. Winter, on the other hand, has more strong edges across the whole scene, while maintaining plenty of weaker edges as well. The snow does seem to have successfully reduced the number of edges on the ground though, as evidenced by the solid black edgeless regions, and it seems more the case that the variance of the laplacian is a poor measure of this.

The variance of the Laplacian in the stationary data (Figure 6.3) mirrors approximately the same average values as videos in motion, increased by between 0.01 and 0.03, which is likely accounted for by the absence of motion blur. The variance has

much less of a range of values in simulated data, which makes sense given that the edges present in the same scene should not been changing much over time.

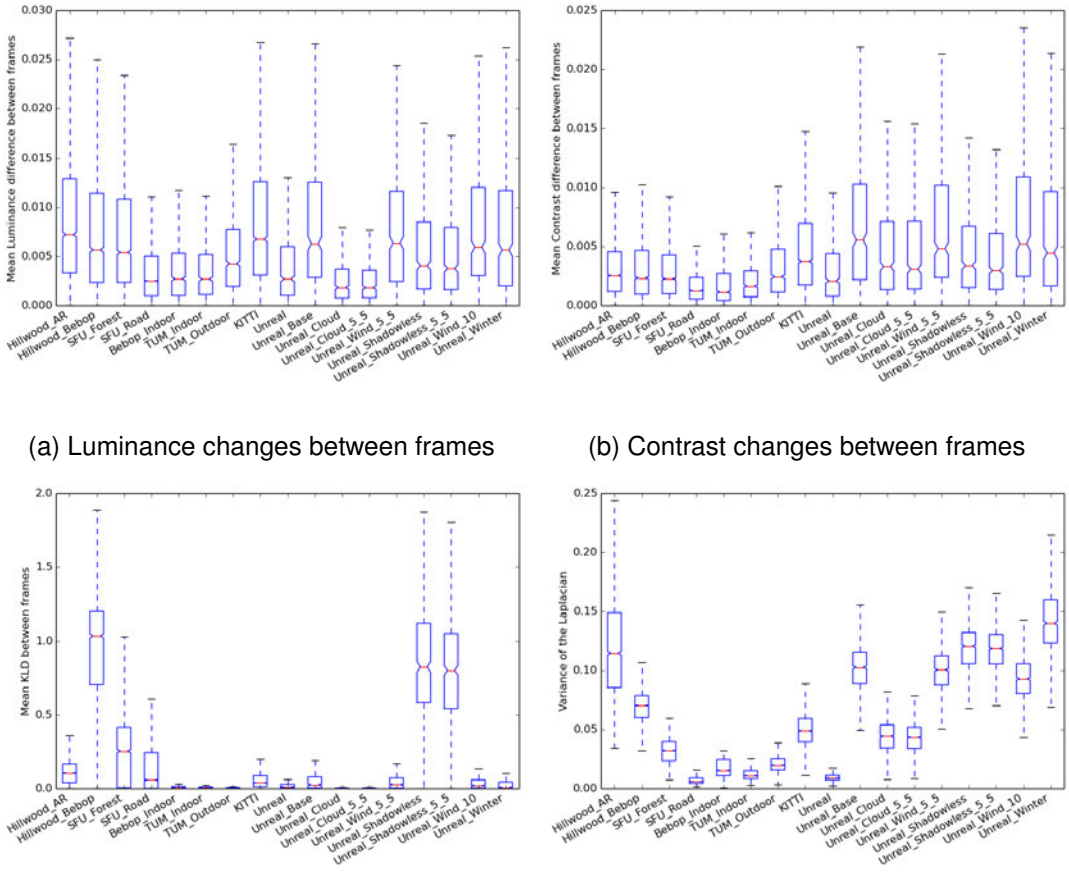
6.3.2 New Simulation Compared To Existing Datasets

Now that we have scene statistics for the new simulated data, we are also able to do a comparison with the original datasets from Chapter 4, where we had highlighted the stark difference between simulated and real forests, to assess whether the new simulated forest data is better reflecting the traits seen in real forests. The stats for the full collection of datasets can be seen in Figure 6.6 and example images from the old and new simulation can be seen in Figure 6.7. Notably there are some differences in the scene statistics presented here for the older datasets than when we first presented them, as the processes have been refined over time, including an update to the framerate for Hillwood AR (which had been incorrectly set as 12 rather than 15).

Luminance and Contrast

We see that luminance change values are now more in line with those for real forest data than in the first Unreal dataset. If we compare images from these two environments side by side, as in Figure 6.7, it seems clear that though the original forest was no more dense, the simulated lighting was not fully reflecting the amount and variability of light filtered through the canopy. Similarly, we can see that the reduction in lighting variation present in the simulated Cloud conditions produces luminance changes more in line with the more consistently lit indoor datasets. Contrast changes appear to have overshot, and where previously the simulation was in line with real data it is now displaying larger contrast changes.

In Figure 6.8 we can see the absolute Luminance and Contrast values, rather than the frame to frame difference. Here the picture is reversed, with the general lighting of the scene as represented by Luminance being lower for all of the simulated datasets than for real data, while Contrast values have a more similar average, but a generally



(a) Luminance changes between frames

(b) Contrast changes between frames

(c) KL divergence between frames

(d) Variance of the Laplacian

Figure 6.6: Scene statistics of the new simulation data, compared against the original set of scene statistics generated in Chapter 4.

wider range. We can conclude that the overall luminance of our simulated forest data is generally lower than real forests, implying a generally darker scene, but that the change in Luminance over time is close to real. On the other hand the average contrast is similar to real forests, but has larger frame to frame changes. A look at the intensity histogram for the simulated Base conditions compared with two real (one forest, one indoor) datasets in Figure 6.9 seems to confirm this observation. While the real data contains a lot of intensity values across the majority of the available range, the simulation is disproportionately skewed towards black, and especially notably has a large percentage of pure black pixels where these are unseen in real data. This skew explains the low overall luminance values, but also potentially high contrast changes: as

Figure 6.7: The first Unreal dataset (Left), compared against the new Unreal Traits (Right).

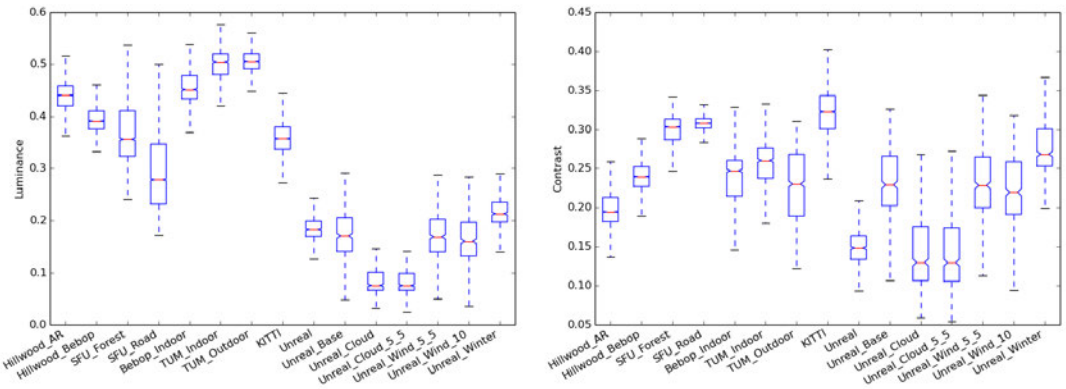


Figure 6.8: Absolute luminance and contrast levels, rather than difference between frames, for each of the datasets.

intensity values being less smoothly distributed implies lots of potential jumps in pixel intensity values, resulting in sudden contrast with their (mostly dark) neighbours.

KL Divergence

The KL Divergence of the simulated data has moved in the right direction from the original simulation, where it was previously more in line with indoor environments, to be now in line with outdoor, but non forested, data. This implies that the simulation is still considerably more predictable than real world forests. One reason for this could be the limited ways in which scene motion is presented by the simulation; leaves move back and forth around a fixed location, but smoothly and predictably, never doing anything like detaching. Also, while leaf litter and other ground detritus is represented

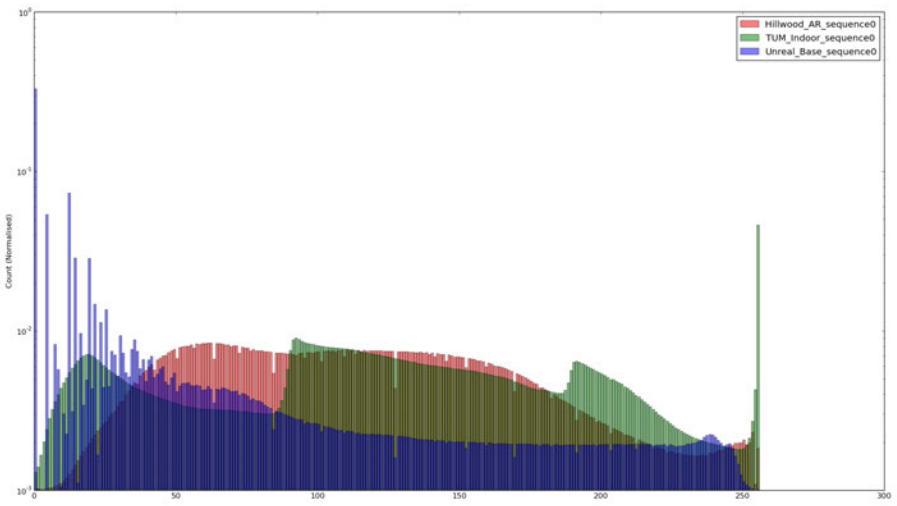


Figure 6.9: Histogram of all pixel intensity values for whole real and simulated video sequences, normalised so that heights sum to 1.

in the simulated scene, these do not move. The combination of these mean that the scene motions are cyclical; everything that moves within the scene does so on a loop, and every point repeatedly covers the same locations. For an algorithm attempting to match points in space over time, this more controlled kind of motion could be a boon, as any previously identified set of points will regularly return to the same locations it previously detected them in. If this behaviour is synchronised over the whole image, this could be a further aid. Of course, given the reservations raised about the usefulness of KLD as a statistic earlier in this Chapter, we should only draw conclusions about increase similarity to real data, not about what that increase actually reflects.

Variance of the Laplacian

The match between simulated Variance of the Laplacian results and those for the real forest data has also greatly improved with the new simulation. Forest data has quite a range of results here, but the simulation was previously very obviously at the far end of the spectrum, showing less edge variation than even the indoor datasets. Now, the

reduced edge return strength seen in Cloud is still in line with lower responses seen for SFU Forest and KITTI, while all other conditions have results more similar to Hillwood AR and Hillwood Bebop, the two highest previous results and those we believe are most representative of "real" forest conditions. Similarly to KLD, we should not draw conclusions about what increased similarity to real data means here.

Shadowless as an alternative to Cloud

We introduced the Shadowless condition after seeing initial results for Cloud and wanting to see if we could create conditions which would reduce sudden lighting changes without other impacts on the appearance of the scene and other statistics. Shadowless presents less strong luminance changes compared to the simulated baseline, as the camera doesn't move in and out of shadowed regions, but the scene is still generally better lit and so varies more than Cloud. Contrast changes in both conditions are approximately the same. Removing shadow appears to have have a large impact on the number of black or dark pixes, with many more intensities over the 50-130 range than other conditions and this could well explain why the KLD for Shadlowless is much higher than other simulations (in line with the previous high KLD dataset, Hillwood Bebop). Finally, Shadowless has raised Variance of the Laplacian compared to the Base, where Cloud was lowered. Again, this is likely due to the wider range of intensity (and therefore generally colour) values present in the scene when the shading is removed. Overall, Shadowless seems like a good potential replacement for Cloud, as it seems to have a closer fit to the statistical properties of real forests and can be used in comparison with Base to see the effect of unreliable canopy lighting on SLAM performance. However, the fact that this is gained by simulating a phenomenon not present in the real world should make us hesitant, in case it has other unintended effects on the appearance of the scene.

6.4 Impact of Environmental Traits on Initialisation

In this section we look at the influence of the controllable environmental traits on the performance of SLAM systems, particularly focusing on the problem of initialisation in ORBSLAM identified by the experiments in Chapter 4.

6.4.1 Ease of initialisation, measured by features required

For the setup of these experiments, we leave ORBSLAM's parameters set as they have been for prior experiments and first focus on how often ORBSLAM can successfully initialise under different environmental conditions with a varying maximum features per frame parameter. We start off using previously explored values of nFeatures, 3000 and 2000, for reliable and unreliable initialisation respectively. We then added two additional test values, one in between (2500) and one easier (4000) as we experimented, to further distinguish between performance in conditions with otherwise similar results. Feature counts higher than 4000 presented a problem for our test system and so were not explored.

Traits	2000	2500	3000	4000
Cloud	0	2	5	5
Base	0	0	4	5
Winter	0	0	3	5
Cloud(5/5)	0	0	0	5
Wind(5/5)	0	0	0	0

Table 6.1: Successful ORBSLAM initialisations (out of 5 trials) with different numbers of features, under varying environmental conditions.

As we can see in Figure 6.1, the different conditions do indeed present varying levels of challenge for ORBSLAM, measured by how high the number of features needs to be in order to initialise. The Cloud condition, which presents low values

for each of the challenging traits (rapid lighting changes, motion unpredictability and complex texture), and would therefore be expected to be least challenging for feature matching (Yang et al., 2018), is as a result the easiest to initialise on. Wind, which was expected to be a challenge, though this is not picked up so clearly in the scene statistics, does indeed prove the most difficult condition. On its own, the motion presented by strong winds stop initialisation even at the highest feature count, and when combined with the easy Cloud condition it is still the second most challenging condition. The only unexpected result is for Winter, which we would have predicted to be easier due to its removal of potentially confusing levels of texture, but which performed at or slightly below the level of the Base condition. There are two potential explanations for this, covered in the above sections. The first is that the ground texture removed by the snow is, unlike in real forests, actually static and therefore a reliable source of features if it doesn't result in too many aliasing problems. Second, as we found when analysing the variance of the Laplacian, while many edges are removed by the snow many of the remaining ones become stronger as a result of increased contrast, and these too may be providing more reliable features.

6.4.2 Impact of pyramid scale factor on initialisation

Starting with the more challenging cases identified, Wind and Cloud, as well as their combination, we ran a separate set of experiments instead looking at the impact of the other parameter found to have had some impact on initialisation performance: pyramid scaleFactor. In these, we used nFeatures values found to have been unsuccessful initialising with the default pyramid parameters, and tried a larger jump between pyramid levels, with corresponding reduction in the number of pyramid levels (nLevels) to keep the smallest level at approximately the same scale. The value chosen, 1.8, was as large as we could go while still have at least 3 levels in the pyramid. When testing nFeatures values that had previously had few or no initialisations with the new pyramid scale, if we had successful results we would then continue to reduce the number

of features again until this performance dropped off, to get an idea for its limits. Finally, we also experimented with changing only the number of pyramid levels and not the scale factor, in case this was what was actually making the difference. The overall

Traits	Features	1.2(8)	1.2(3)	1.8(3)
Wind(5/5)	4000	0	0	0
Wind(5/5)	2500	0	1	0
Cloud(5/5)	3000	0	5	5
Cloud(5/5)	2500	0	5	5
Cloud(5/5)	2000	0	5	3
Cloud	2500	0	5	5

Table 6.2: Successful ORBSLAM initialisations (out of 5 trials) with different numbers of features and pyramid structures (displayed as scale between levels with total number of levels in brackets), under varying environmental conditions.

finding from this set of results (in Figure 6.2) is that changes to the pyramid structure clearly support initialisation on otherwise challenging datasets, reducing the number of features required in both clouded conditions. With the scaleFactor increased to 1.8, Cloud(5/5) is able to initialise 3 out of 5 trials with half as many features as were previously needed (2000 vs 4000) and reliably initialises at 2500. Cloud reliably initialises with 2000 features where previously it needed 3000 for this. Pure Wind(5/5) still fails to initialise even at 4000 features, however, and attempting to increase this any further is beyond the capabilities of the hardware.

Interestingly, we see the same level of improvement if not better in the control case where we only reduce the number of scale levels and not the factor, resulting in the only successful initialisation we have seen on Wind(5/5). This implies that either it is not specifically *which* pyramid levels are being used that matters so much as how many, or that the levels having a negative effect on initialisation are the ones which are less present in both cases: the low resolution ones. As the number of features to be

extracted is divided over the pyramid levels, reducing the number of levels has effectively the same impact as increasing the number of features on each level. We can infer two conclusions from this that are not in contention with each other: the first is that the high resolution layers of the pyramid are more useful for successful initialisation when sufficient features are extracted, and the second is that the absence of lower resolution parts of the pyramid doesn't seem to impede initialisation. Future work could establish whether this continues to be true in tracking past initialisation, but it could be that a focus on the precise high resolution features is specific to initialisation's need to establish an accurate baseline.

6.4.3 Successful feature matches between frames

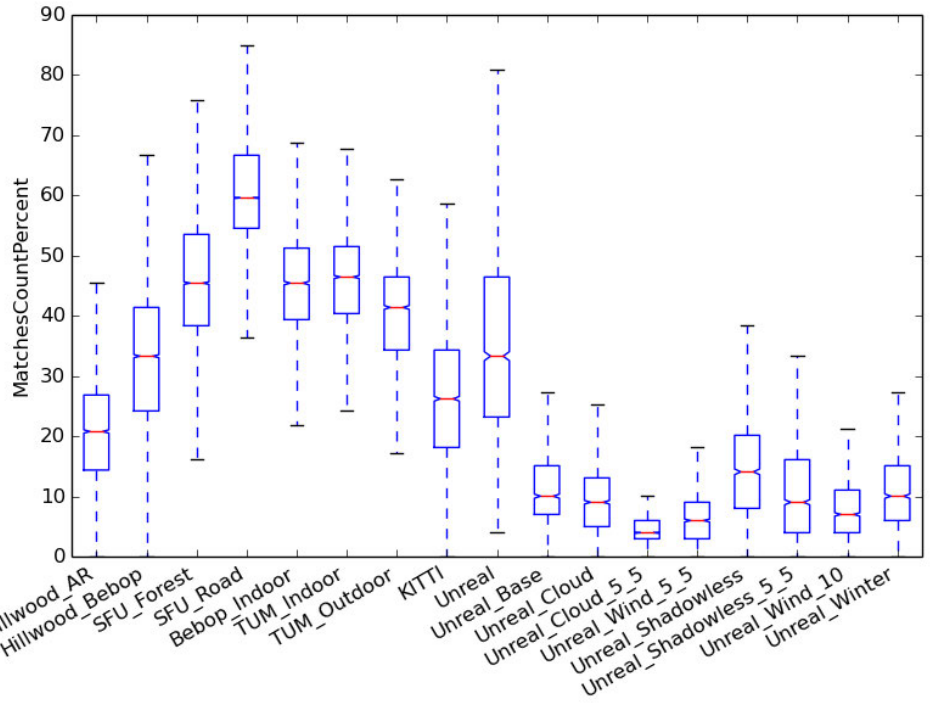


Figure 6.10: Successful ORB feature matches (as a percentage) for each of the datasets.

As a further final investigation, we generated frame to frame feature match success rates, as in Chapter 4, but using ORB features instead of SIFT to be more similar to

ORBSLAM of interest. These results are shown in Figure 6.10, and generally match up with the initialisation performance, with Winter having approximately the same match rate as Base, and Wind having lower. The interesting result is for Cloud, which not only doesn't display an increased matching rate but also, when combined with wind in Cloud(5/5), worsens rather than improves it over the rate for Wind(5/5). It is possible that something ORBSLAM adds over the base ORB matching process (such as the pyramid levels or forcing features to be distributed over the whole image) improves initialisation performance particularly in low light conditions. The other result of note here is that the updated simulation has lower feature matching rates than any of the real world data, where previously we had found the static simulation unrealistically easy to match against. It is reassuring that attempting to use the scene statistics as a guide seems to have successfully introduced more of the challenge of real world data, but in its current state we have overshot and more work is needed to close the gap is needed in future.

6.5 Conclusions and Future Work

We set out in this Chapter to further investigate the relationship between our scene statistics, the environmental traits (such as wind) that we are using them as measurements for, and performance of SLAM systems in environments with varying traits. Our previous work in Chapters 4 and 5 had analysed the link between the scene statistics and SLAM performance, but not how well the statistics actually reflect the environmental traits we expect them to, or how the traits impact performance, so these were both explored here.

In order to have control over varying environmental traits we extended our existing simulation with environmental conditions and created a pipeline to allow for easy recording and replay of routes. We recreated conditions within the simulation which we expected to involve a substantial expression of particular environmental traits. We

were then able to record the same route in each set of conditions, enabling us to assess both how these impacted statistical measurements and SLAM performance.

We measured the changes to scene statistics presented by the varying simulation conditions, to evaluate their ability to reflect particular traits of the scene like lighting or motion. Lighting related statistics, Luminance and Contrast, are generally seen to behave as expected in Clouded or Shadowless conditions, which remove the effect of inconsistent lighting through the canopy. We find that Cloud's general reduction of lighting in the scene comes with a knock-on impact on the range of intensity or colour values, which we posit also explains unexpectedly low KLD and Variance of the Laplacian measurements. Shadowless has a much less notable effect on Variance of the Laplacian, but a large effect on KLD in the opposite direction, as well as a less realistic an appearance. Wind is shown to have an impact on KLD (through scene unpredictablility), though only notable when the camera is static. The implication here is that motion caused by the camera moving through the environment has a strong impact on KLD than the in-scene motion we wanted it to measure. Overall, these results present a strong case against using KLD for this purpose, except where confounding factors can be controlled for. Finally the Winter condition, meant to reduce the Variance of the Laplacian in the scene by reducing the texture, had the opposite effect. The increase in strong white-to-black edges produced by this condition seems to have increased the variance, revealing a limitation of this scene statistic: that "amount" of texture is a challenging concept to measure, and involves strength as well as number of edges. This invites follow-up work, to look at whether we can identify the difference between scenes with strong or numerous edges, and how the trade off between these influences SLAM performance.

Comparing the new simulation's statistics with real world datasets, we find it is generally more closely reflecting realistic traits than before, but with some room still for improvement. Firstly, the simulated data has unusually high contrast changes, and does not display levels of KL Divergence as high as real forest data. We trace one

potential cause of this to a restricted colour palette and particularly heavy use of pure black by the simulator. Secondly, the Winter condition appears to be a poor way to simulate a low texture version of the environment, as it strengthens some existing edges while removing others. This is as measured by Variance of the Laplacian, however, and as noted above this has limitations as a measure. Intuitively it still seems as though there should be *less* texture present in the Winter scene, and for SLAM systems that still may well make a difference even if the edges remaining are *stronger*. Future work should produce a alternative low texture version of the simulated scene with ground litter textures removed, as well as take advantage of the feature added in recent GmbH (2022) asset packages that allows removal of foliage from trees. Thirdly, the simulation has static ground texture (eg. leaf litter that doesn't move), which could be providing a lot of very reliable texture on the ground when compared to leaf litter and low grasses in real forest data.

Overall, some traits displayed by the simulator have grown closer to real world forest data, indicating that we have made some progress in closing the sim to real gap and that this simulation is becoming a more viable tool for testing a SLAM system's robustness to the challenge of real forest environments. The additional interfaces added to the simulation for recording and replaying routes, which can include ground truth data, also open up the opportunity for more in depth SLAM development in future.

Finally, we investigated how the performance of ORBSLAM (focusing on initialisation) is impacted by the varying simulated environmental conditions. These experiments confirmed most of our expectations about the challenge of different environmental traits; reductions in lighting changes in the Cloud conditions made initialisation easier, and extra in-scene motion caused by the Wind conditions made it more difficult. The Winter condition did not make initialisation easier as we had expected it to, but given that our analysis has also brought doubt upon this as a method of reducing texture we cannot draw any strong conclusions about how low texture impacts performance from this. Performing feature matching experiments directly, we find that

all of our simulated data is now more difficult than real data to match features on, and so while the realism of our simulation's ability to reflect challenging natural traits has been improved, it is still not recommended as a complete replacement to development of SLAM using real world data, where that is possible. Since this work was completed, Wang et al. (2020) performed a study of SLAM performance over a wide range of simulated environments, including some forests, and although they not go into the same depth looking at the traits of those environments they come to the same conclusion as us about SLAM: that adaptability between domains is far from solved.

Chapter 7

Preprocessing for Forest Environments

7.1 Introduction

The previous work in this thesis has found that of the two major components, tracking and loop closure, the former appears to be the more salient limitation when adapting to challenging environments such as forests. Chapters 4 and 6 identified measurable traits of forest scenes which are likely to contribute to their difficulty, and Chapter 5 confirmed that ORBSLAM struggles to match features between two views of the same forest scene. This chapter investigates whether early processing of video input data, loosely inspired by biological visual systems, can compensate for those difficult traits and enable better performance in frame-to-frame tracking.

The visual scene statistics identified in Chapter 4, and used throughout this thesis, were inspired by the work of visual ecologists studying animal vision. Their assumption is that animal eyes and brains are adapted to efficiently represent natural scenes, and so they measure the traits of these scenes in order to attempt to correlate them with the adaptations of visual systems. This in turn produces an understanding of what processing done by visual systems might be contributing to addressing certain challenges

of perception.

Other works have taken inspiration from nature to improve visual navigation systems, such as using optical flow response for flight control (Moore et al., 2010; Sabo et al., 2016), UV sensitivity for extracting skyline features (Stone et al., 2016) or low resolution for robustness against noise and motion (Milford, 2013).

The experiments described here take advantage of the EnVisionEd modular setup described in Chapter 3.3 to prototype visual pipelines performing simple preprocessing and evaluate the impact on the frame to frame feature matching task on forest datasets. Outputs from these processing pipelines are then also input to the SLAM benchmarking tool SLAMBench 2.0 (Bodin et al., 2018), which allows for more precise trajectory accuracy benchmarking.

7.2 Materials and Methods

7.2.1 Datasets and Scene Statistics

The datasets chosen for this work were primarily chosen to exemplify various levels of challenge in forested environments. SFU Forest and Road are recorded from the same ground vehicle on a track surrounded by trees, where the difference between them is that Forest is under canopy cover. Hillwood AR and Bebop are our own original recorded UAV forest datasets, with different resolution and frame rate, which we know from previous work are particularly challenging for SLAM systems. Forest Loop is a newer route recorded on the same Bebop setup with careful movement to try and rule out camera motion as a limitation. Finally ICL NUIM (Handa et al., 2014) is a synthetic indoor dataset used by benchmarking software SLAMBench to evaluate the quality of route and scene reconstruction. The key information about all datasets is in Table 7.1.

Note: this work was completed before that presented in Chapters 5 and 6, so Forest Loop 2 (with ideal initialisation) and the improved simulation environment had not yet

been created.

Table 7.1: Datasets used in this work, with video count, frame rate, resolution and an example frame.

Name	SFU Forest	SFU Road	AR	Bebop	Forest Loop	ICL NUIM
Count	4	4	3	1	2	4
FPS	30	30	12	30	30	30
Res	752x480	752x480	320x240	1920x1080	856x480	640x480
Eg.						

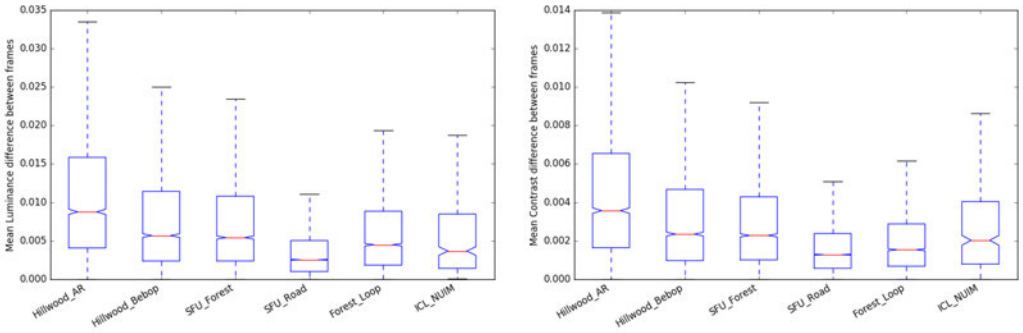
In Figure 7.1 we present the Scene Statistics, as calculated in Chapter 4, for the ICL NUIM dataset for comparison against the other datasets being used. It presents luminance and contrast changes marginally lower than the forest data, similarly to how the real indoor dataset (TUM Indoor) did. Its results for KLD between frames are unusual, as they suggest very high in scene motion when there is practically none in this dataset.

7.2.2 Early Processing

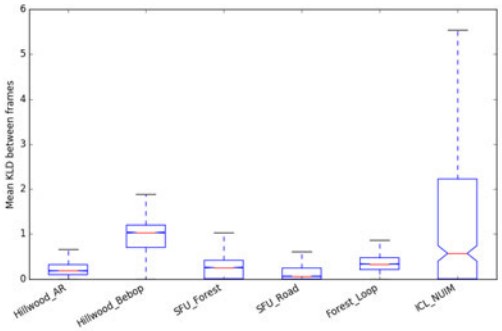
Our tests involve applying one or more modifications to the input data frames before passing them on to our test algorithms. We refer to these modifications as "early processing steps". They are sub-divided into categories, which are elaborated upon in this section.

7.2.2.1 Colour Channels

Animal visual systems have different receptors for different points on the visual spectrum, and the number and tuning of these varies between species. Human eyes have receptors sensitive to red, green and blue regions of the spectrum, while honey bee



(a) Luminance difference between frames. (b) Contrast difference between frames.



(c) KL Divergence between frames.

Figure 7.1: Scene statistics for ICL NUIM dataset, presented alongside those previously calculated for other datasets.

ommatidia contain receptors for green, blue and ultraviolet (Wakakuwa et al., 2005). Receptors are also not present in equal numbers, both humans and honey bees having majority green receptors, meaning that a visual system can be heavily tuned towards a particular colour.

The sensitivity of visual systems towards different colours may reflect on the useful information found at those wavelengths in natural scenes. If the natural world contains a large amount of green objects then it may be more critical to have sensitivity to differences between them in order to distinguish nearby plants against a background of other plants. Likewise, some wavelengths may be less reliable due to being noisier under some conditions, as other works have suggested blue is in low light (Zhang et al., 2012; Aladem et al., 2018).

Modern visual SLAM techniques have overwhelmingly settled on using monochrome images as their inputs, with the conversion to grey scale by averaging the three colour channels being the first step many systems perform on their input data. This has the benefit of reducing computational complexity early on, and many of the features these systems are looking for (such as corners) remain visible in intensity images.

The sensors and data formats used for visual datasets restrict the colour spaces available to this work, but within these it is still possible to investigate how images from single colour channels and pairs of colour channels perform compared to the baseline intensity image.

7.2.2.2 Regional Focus and Resolution

Concentrations of receptors vary across the visual field in many animals, such as the concentration of cones at the human fovea (Jonas et al., 1992), ommatidia along the "foveal belt" of desert ants (Zollikofer et al., 1995), or ommatidia along the horizon for fiddler crabs (Layne et al., 1997). These imply an in built focus in these visual systems for information in particular regions of the visual field. Central points of receptor density tend to be associated with predators or other behaviours that require

focusing on detail in a small region (as with dragonflies or humans), whereas horizontal regions of density are proposed to be related to navigation tasks, when information at the horizon maybe be most informative.

Varying pixel density across an image would require a new file format and processing, but we can investigate the relative information of different regions of an image by dividing it into overlapping top, middle and bottom regions (each 50% of the image) and testing these against each other.

Many animals (such as insects) also have visual systems with much lower total receptor count. Honey bees, for example, have approximately 11000 ommatidia in total (Somanathan et al., 2009), compared to the 76800 (320x240) of our lowest resolution video dataset. Smaller images can be more effective for image matching (Milford, 2013) and so we investigate the influence of this here with a "resize" processing step that scales down each of the axes by half.

7.2.2.3 Filters

Local connections between cells in animal visual systems sometimes lead to various smoothing effects, which remove noise from images. These can be approximated by filters applied to the image in our pipeline.

Gaussian filters, which replace a pixel value with the weighted average of its neighbours, are a commonly used choice for removing noise from an image but carry the disadvantage of also smoothing other details such as edges. A Bilateral filter takes into account not just the proximity of other pixels in space but also the similarity of their original values, and therefore preserves edges more effectively. Finally, the Wiener filter has been proposed to be analogous to some early visual processing (Simoncelli and Olshausen, 2001), capable of removing both noise and blur from an image and adapting to luminance levels.

7.2.2.4 Adaptive Histogram Equalisation

Histogram equalisation normalises and improves the contrast in the image by spreading out values over the available range. It has been shown by Laughlin (1981) and Srinivasan et al. (1982) that the response of bipolar cells in the eyes of flies perform histogram equalisation, and integrating contrast normalisation into feature detection has shown promise before (Vonikakis et al., 2012). Here we choose to use Adaptive Histogram Equalisation (AHE) as this normalises local contrast, more similarly to the eyes. Furthermore we use Contrast Limited AHE in order to avoid over-amplification of noise.

7.2.2.5 Summary

Processing pipelines have been created for applying the following to datasets:

- Colour Channels: Red, Green, Blue, Cyan, Yellow, Magenta
- Filters: Wiener, Gaussian, Bilateral
- Image Regions and Resolution: Top, Middle, Bottom, 50% Resize
- Contrast Limited Adaptive Histogram Equalisation (CLAHE)

7.2.3 Feature Matching

The first test algorithm evaluates the performance of a feature matcher outside of the complex context of a full SLAM system. We choose to use ORB features due to their balance between descriptive power and speed of processing, as demonstrated by their use in ORBSLAM (Mur-Artal and Tardós, 2017). The method is the same as in Chapter 4. Features are extracted in each pair of subsequent frames, the matcher attempts to find matches that pass a ratio test (as described in Lowe (2004)), and the percentage of features successfully matched is reported.

This method is a simplification of the frame to frame tracking process in any complete SLAM system, where other processes such as feature pruning and geometric consistency checks are often in place to improve the reliability of feature detection and matching. It has, however, been found to act as a reliable baseline measure of the performance on a given set of data.

7.2.4 Benchmarking

The second test algorithm utilises SLAMBench 2.0 (Bodin et al., 2018) as a robust tool for benchmarking the performance of SLAM systems. SLAMBench provides a suite of tools for quantitative testing of SLAM systems, including trajectory and scene reconstruction accuracy. ORBSLAM, the focus of this work, is provided as one of the pre-integrated SLAM systems.

The down side of this approach is that it limits us to using datasets integrated into SLAMBench, and as the integration requires ground truth data that doesn't exist for forest datasets we cannot use our ideal datasets. Instead what testing with SLAMBench allows is analysing performance changes when each of our processes are applied to a more standard indoor SLAM simulation dataset, ICL NUIM. The dataset consists of 4 trajectories through a simulated living room, following progressively wider sweeps of the room. This environment should benefit less from the changes and therefore be a more challenging test case.

To apply the preprocessing, we utilise the same configuration files as for the default version of the ICL NUIM videos, and simply replace the frame files with the preprocessed versions, then run SLAMBench's performance evaluation tools as normal. We are also limited in which early processing steps can be tested, as anything that changes the size of the image would stop the integration with calibration and ground truth from working. The outputs of this benchmark test give us both Absolute Trajectory Error (ATE), which is a good measure of the consistency of the whole trajectory, and Relative Pose Error (RPE), which is a better measure for local accuracy of trajectories

(Sturm et al., 2012). We run the benchmark on each trajectory 3 times.

Doing this work we discovered an issue with the monocular ORBSLAM 2.0 evaluation tools in SLAMBench, and so the results presented here are from RGBD tests instead. The added depth information would be expected to make up for poor tracking performance in some situations and generally result in overall greater accuracies, but we would expect the relative impacts of visual appearance traits to still be discernable.

7.3 Results

7.3.1 Feature Matching

7.3.1.1 Dataset Baselines and Trends

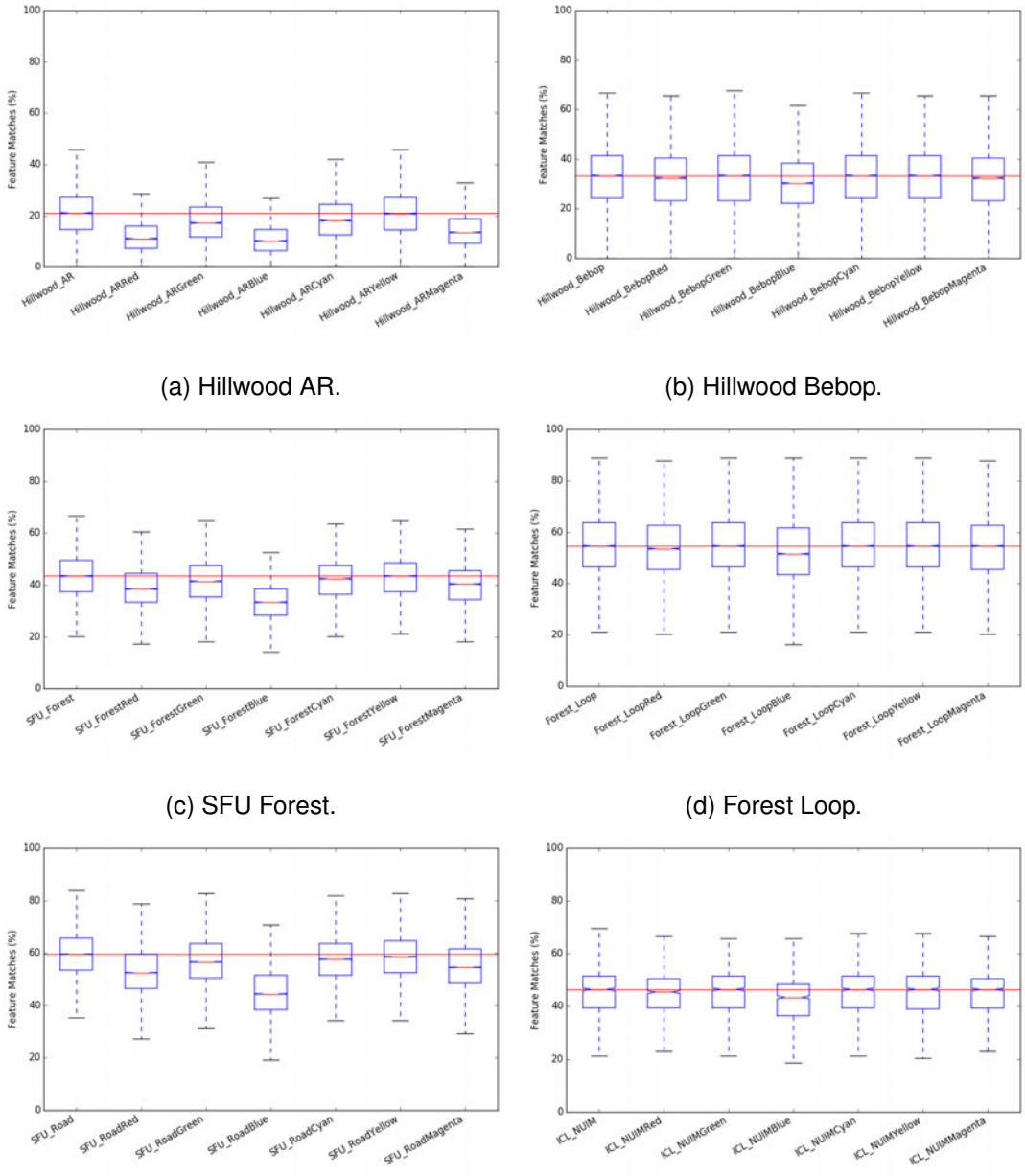
The datasets tested show a variety of baseline matching performances (the first box on each plot). The best results come from SFU Road, with a baseline match success average of 60%, followed by Forest Loop in the 50s, SFU Forest and ICL NUIM in the 40s, Hillwood Bebop in the 30s and finally worst results from Hillwood AR averaging just above 20%. The overall changes preprocessing steps make are generally pretty small. The best cases are improvements of around 10% while the worst cases drop by up to 20%.

The worst average matching performances on the two Hillwood datasets are accompanied the worst minimum match performance; these are the only datasets where the matching algorithm fails to find any matches between some pairs of consecutive frames. This is likely due to these being the two videos with the most erratic camera motion. Sudden movements of the camera could minimise the amount of the seen visible between two frames, or at least blur it to the extent that it was difficult to detect features.

Forest Loop was created as a follow up to Hillwood to address the challenging camera motion, and this seems to have been successful. In addition to one of the

higher averages, this dataset shows higher minimum match percentages and the highest maximums of all of the test datasets.

7.3.1.2 Colour Channels



(a) Hillwood AR.

(b) Hillwood Bebop.

(c) SFU Forest.

(d) Forest Loop.

(e) SFU Road.

(f) ICL NUIM.

Figure 7.2: Feature matching success rates for datasets and combinations of their colour channels.

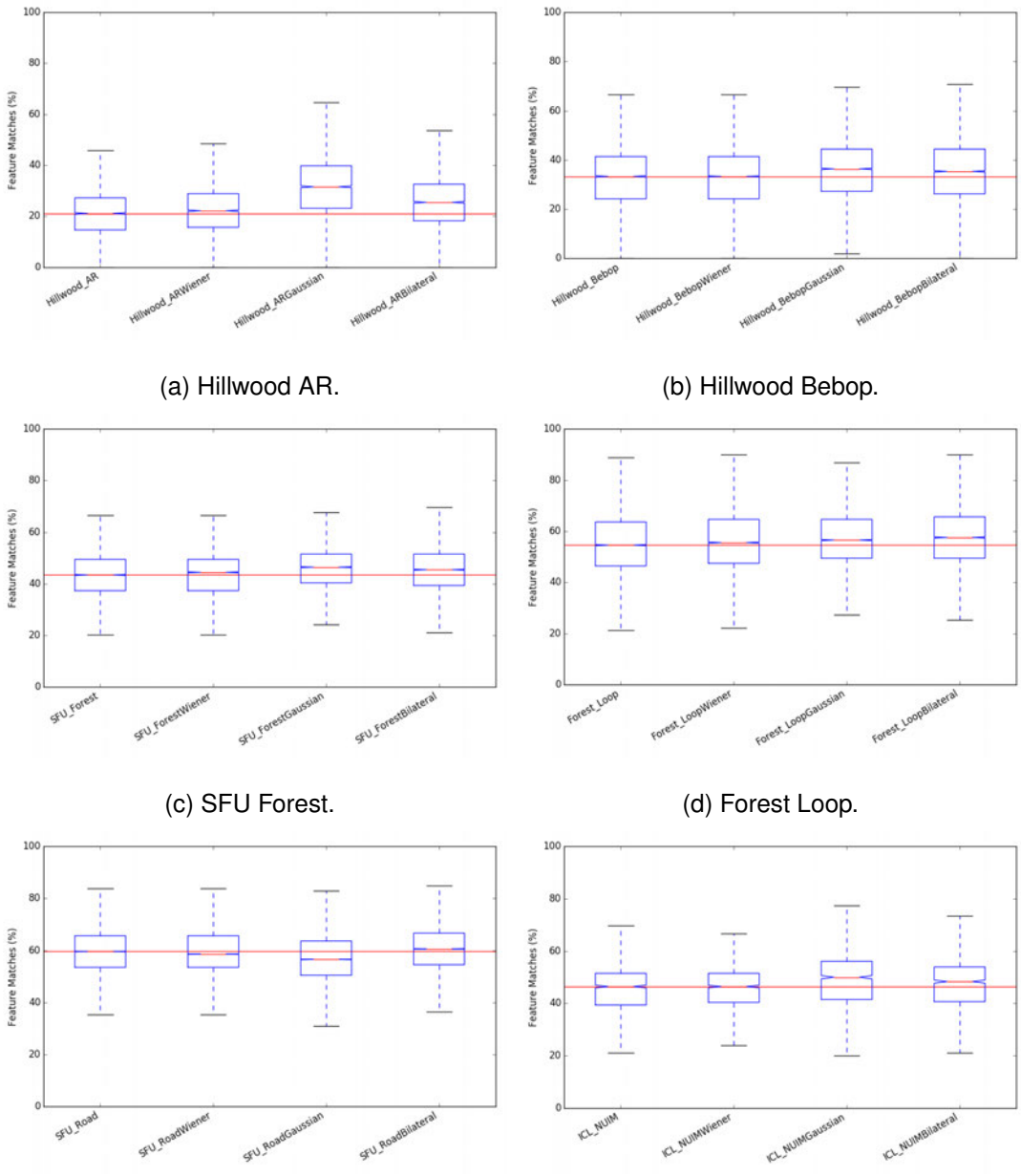
For the primary colour channels in Figure 7.2 we see minimal difference in Hill-

wood Bebop, Forest Loop (also recorded with the Bebop) or ICL NUIM. The trend in the other datasets is similar across the three of them. The median match percentage for Green is always higher than for Red, which likewise always has a higher median than Blue. In these three datasets, this difference is particularly pronounced for the Blue channel, which also performs notably worse than baseline. Bebop recordings and ICL NUIM displaying minimal differences suggests that the effects do not correlate with the environment but perhaps more with the sensor, and visual inspection of the separate channels for Bebop data does confirm that very little difference is visible, suggesting some normalisation is occuring. We see a very similar trend in minimum and maximum values, though less consistently.

In the three datasets with notable variation, the pairs of colour channels reliably match or outperform the individual channels making them up. Yellow (Green and Red) matching the baseline, implying that the Blue channel provides little to no additional information. Magenta (Red and Blue) appears to perform worst, and does not outperform single channel Green. Cyan (Green and Blue) is between the other two, but does beat pure Green.

7.3.1.3 Filters

The three filtering approaches perform at or above the baseline in the majority of cases in Figure 7.3, but in almost all cases the difference is not enough to conclude any improved performance, though it does seem that none of these approaches negatively impact performance either. The exception is the Gaussian filter on HillWood AR, where on average it is achieving 50% more matches between frames than the baseline. This dataset is particularly noisy with leaf litter and small camera motions where the Gaussian filter, much in the same way as resizing the image (Milford, 2013), might smooth over these confusing small scale details to allow for better matching.



(a) Hillwood AR.

(b) Hillwood Bebop.

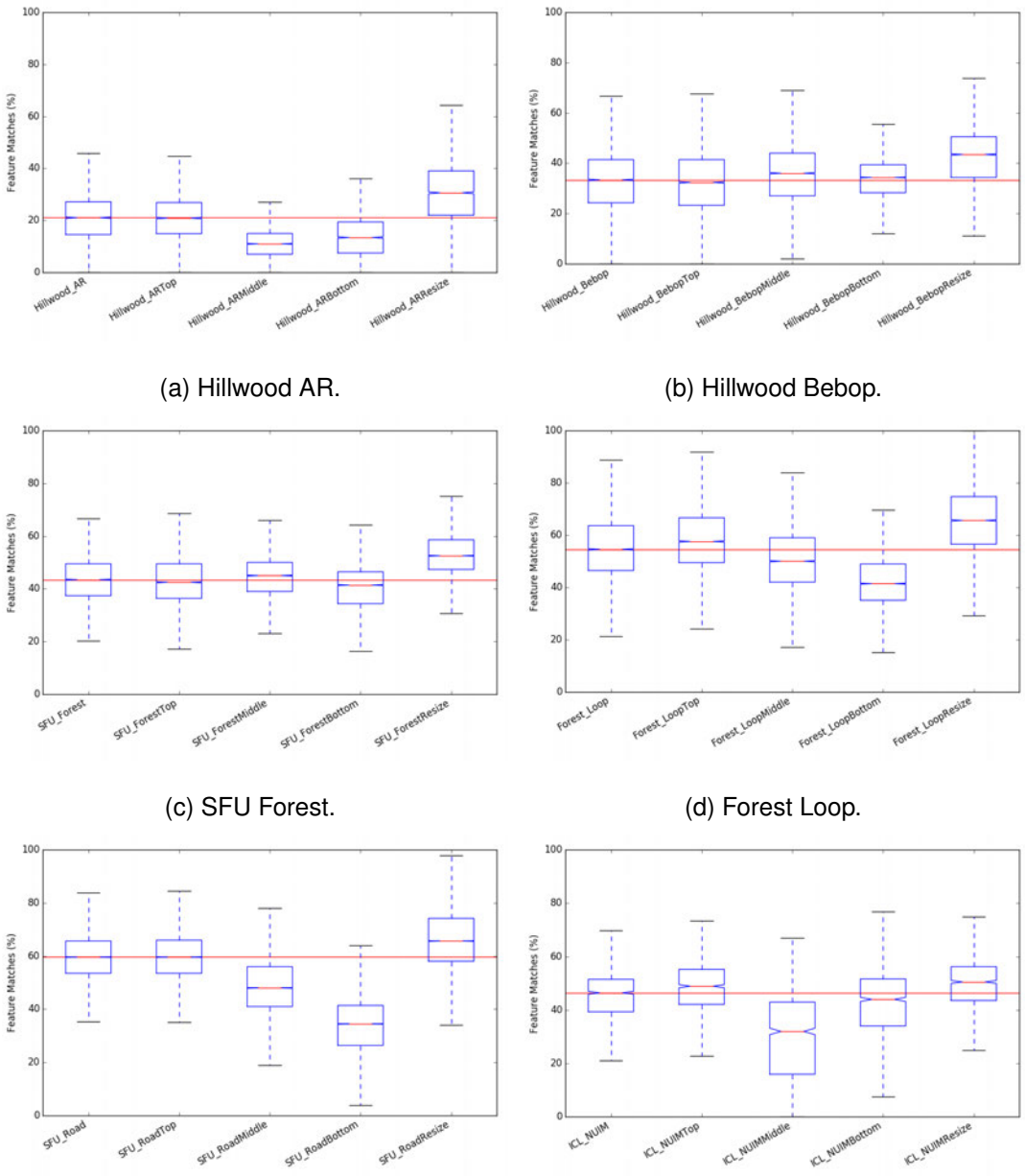
(c) SFU Forest.

(d) Forest Loop.

(e) SFU Road.

(f) ICL NUIM.

Figure 7.3: Feature matching success rates for datasets and versions of them where a filter has been applied.



(a) Hillwood AR.

(b) Hillwood Bebop.

(c) SFU Forest.

(d) Forest Loop.

(e) SFU Road.

(f) ICL NUIM.

Figure 7.4: Feature matching success rates for datasets and sampled subregions of them, including a 50% resize.

7.3.1.4 Regional Focus and Resolution

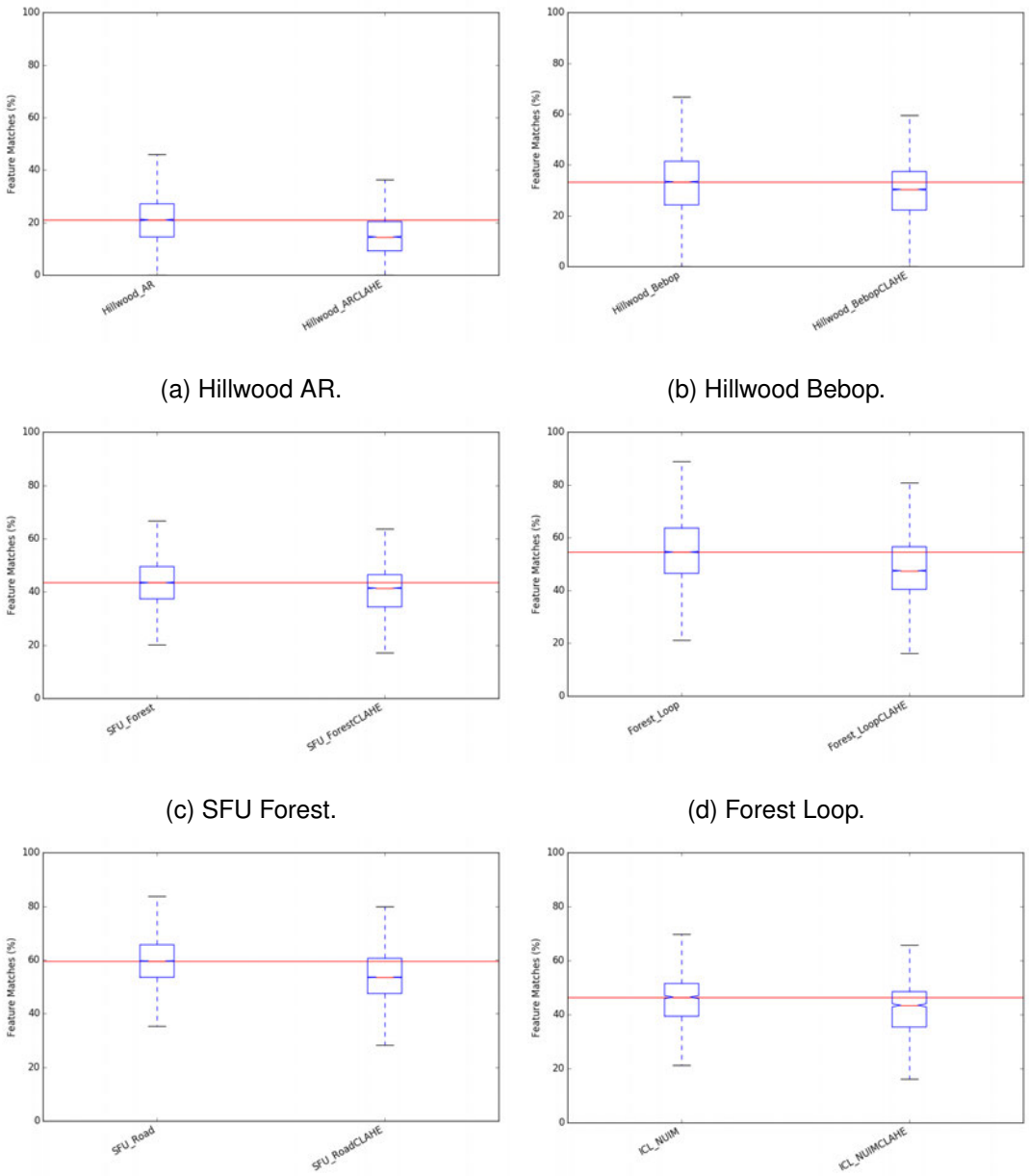
Restricting to sub-regions of the field of vision does not improve the matching performance as notably as resizing in Figure 7.4, but as with colour channels we are equally interested in the distribution of useful information. In all of the datasets, Top is performing indistinguishably from the baseline, in-keeping with previous work by Stone et al. (2016) showing the skyline is informative for navigation. Middle and Bottom are both substantially worse than baseline in half of the tested datasets. Perhaps notably, due to the angling of the camera, Middle in Hillwood AR and ICL NUIM is also pointed at a downward angle, and so these results generally point towards matching on the ground being more challenging in most environments.

On SFU Road, the poor performance of Middle and Bottom while Top matches the baseline is to be expected from the hypothesis that all of the useful information is on the skyline. When the platform moves under canopy for SFU Forest, the distinctive skyline features are lost and the surroundings become more varied, so the locations of useful features spread out across the scene.

Resizing the input images results in an increase in matching performance in all datasets, although this is much less pronounced in ICL NUIM. This could be due to the effect of resizing in reducing complex textured areas more present in forest data than in indoor simulation. The only dataset where resizing did not have the best performance was on Hillwood AR, where Gaussian blurring performed approximately as well. As AR and NUIM are the two lowest resolution datasets to begin with, it is likely that the less pronounced results seen on these are due to diminishing returns from low resolution.

7.3.1.5 Adaptive Histogram Equalisation

It can be seen in Figure 7.5 that CLAHE makes no substantial difference to performance compared to baseline in any dataset except Hillwood AR, where it is worse. There is no obvious explanation for this, except that the normalisation is interfering



(a) Hillwood AR.

(b) Hillwood Bebop.

(c) SFU Forest.

(d) Forest Loop.

(e) SFU Road.

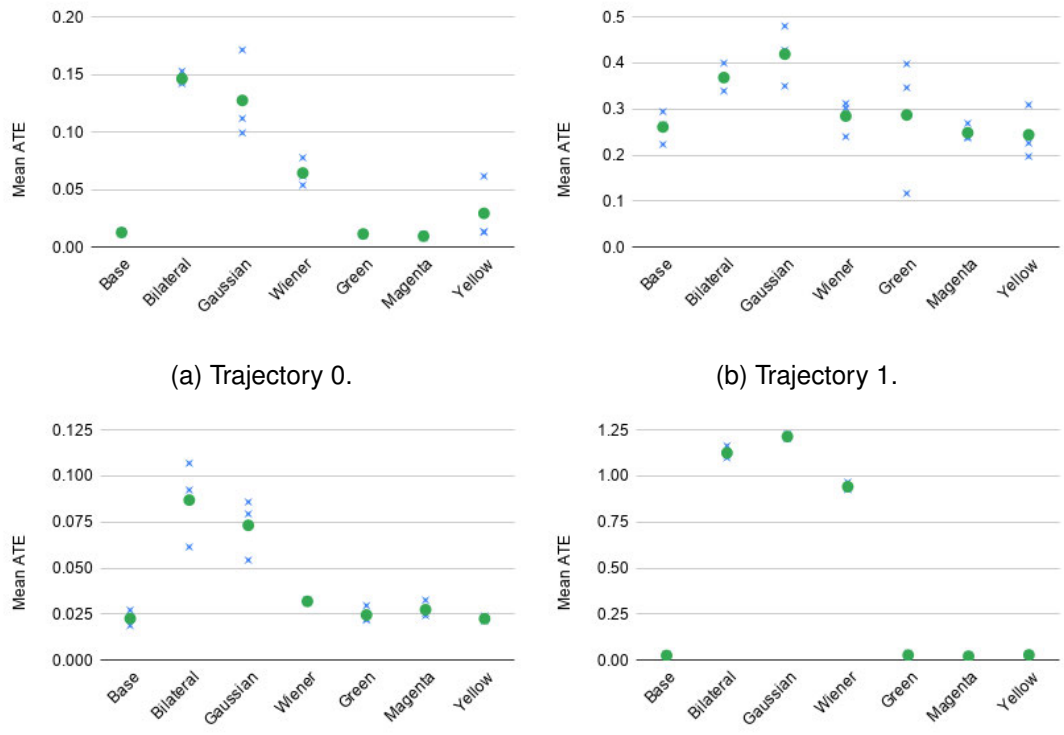
(f) ICL NUIM.

Figure 7.5: Feature matching success rates for datasets and a version of them that has had Contrast Limited Adaptive Histogram Equalisation applied.

with how features are detected or described in an unexpected manner.

7.3.2 Benchmarking

Benchmarking provides more accurate performance evaluation of how a full SLAM system would perform under real world conditions, but at the expense that the choice of evaluation must be fit to the limitations of the benchmarking tool, as outlined in Section 7.2.4. Here that limits us to comparing ORBSLAM's performance on a simulated living room dataset, and only using Filter and Colour Channel preprocessing steps, but this should still provide a useful insight into the performance of the system. We also choose to limit Colour Channels to those including the Green Channel, as these performed best in the feature matching.



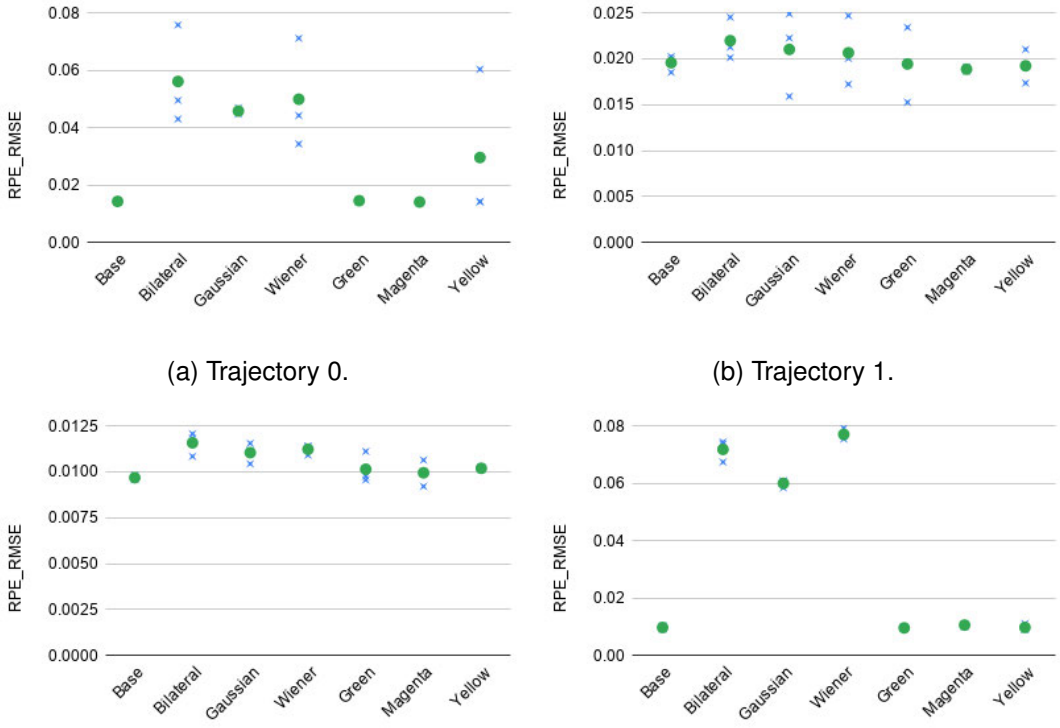
(a) Trajectory 0.

(b) Trajectory 1.

(c) Trajectory 2.

(d) Trajectory 3.

Figure 7.6: Mean Absolute Trajectory Error of ORBSLAM 2.0 pose estimate on each of four ICL NUIM trajectories with various preprocessing steps applied. Each test has been run three times and a green dot indicates the mean.



(a) Trajectory 0.

(b) Trajectory 1.

(c) Trajectory 2.

(d) Trajectory 3.

Figure 7.7: Relative Pose Error of ORBSLAM 2.0 pose estimate on each of four ICL NUIM trajectories with various preprocessing steps applied. Each test has been run three times and a green dot indicates the mean.

Similarly to the feature matching results, single colour channels generally do well in these benchmarking tests both in terms of ATE (Figure 7.6) and RPE (Figure 7.7). Colour error values do not generally differ much from the baseline, although Yellow struggles on Trajectory 0, where by both error metrics it has a worse average and wider distribution than the other colours and baseline. Trajectory 0 is the only route that never looks towards the end of the room lit from 'outside' by natural light, so it is likely that this is reflective of the Blue channel containing less information in the 'indoor' lighting.

Unlike on the feature matching task, filters often notably worsen the accuracy of trajectory estimation. For Trajectories 1 and 2, the poor filter performance is limited to the Gaussian and Bilateral filters, and then only in terms of ATE not RPE. As ATE is more sensitive to accumulation of error, this would imply that these filters contribute to an error early but perform well for the rest of these trajectories. One possible explanation is that 1 and 2 look at the walls and ceiling more at the start of the videos, while in 0 and 3 it is more spread out. The walls in the simulated living room have slightly textured wallpaper, which might provide reliable features under baseline (or colour channel) conditions, but might easily be lost to blurring effects from filters.

7.4 Conclusions and Future Work

We presented here a set of experiments investigating the use of preprocessing steps applied to visual datasets before visual odometry. Inspired by animal early visual systems, it was hoped that some of these modifications would compensate for difficult traits of natural scenes, such as dynamic lighting and texture complexity. Due to limitations of time and available datasets, we could not perform a full benchmark on real forest data, so instead we assess feature matching on forest data and perform a full benchmark on simulated indoor data.

We find that the Green colour channel performs roughly as well as the baseline

for feature matching in most cases, as well as consistently matching or outperforming the other colour channels. Likewise for the benchmarking tests, Green performed indistinguishably from the baseline. These results support the idea that plenty of useful information for localisation might be found in the green part of the spectrum in forest scenes, as reflected in the strong preference towards this range in many organisms' visual systems. In some memory-restricted instances it is useful to know that the green channel might be sufficient on its own, but there doesn't seem to be any evidence for making this restriction if space isn't an issue. In animal visual systems, colour processing can be oppositional, benefitting from comparing one or two channels against another to determine useful information, and future adaptations of this work could investigate this approach.

The filtering processes tested also match, but mostly don't exceed, the baseline in the feature matching task, though there is no obvious use for this as they don't improve memory efficiency. The Gaussian filter provides the only notable improvement, increasing the average matches by 50% on the most challenging Hillwood AR dataset, suggesting that it may have particular usefulness in complex texture scenes where it may be smoothing out smaller, less reliable features. Notably though, the filtering processes generally worsen the benchmarking performance. Whether this is due to these scenes having a lot less texture complexity or due to an unexpected interaction between filtering and the depth data, it seems as though there is little evidence to suggest filtering can be useful to SLAM unless the conditions the Gaussian aids with in Hillwood AR can be identified.

Finally, we tested the effect of restricting the matching area of the image to certain sub-regions. These mostly either meet the baseline or are noticeably beneath it. Either the top or middle half of the image seems to contain the most useful information in all cases, meeting or slightly exceeding the baseline, likely dependent on the angle of the camera. These could not be tested on the benchmark, but the results do suggest that a smaller field of vision limited to the upper region could be used in some applications

to reduce image storage size and speed up processing. A more adaptive approach could be taken, even, if scene-appropriate feature detection regions could be selected algorithmically, such as by utilising an attentional neural network.

Chapter 8

Place Recognition in Forest Environments

8.1 Introduction

As we noted Chapters 4 and 5, loop closure appears to be one aspect of SLAM where existing state of the art systems struggle when presented with unstructured natural scenes. If a robot cannot recognise when it has returned to a previously visited place, then it will struggle to compensate for the drift that is inevitable when run over any significant time or distance.

This chapter presents our further investigation into the performance of both algorithmic and deep neural network based approaches to loop closure, and how well they generalise to the conditions presented by forest environments.

The standard algorithmic approaches compared are Bag of Words (Gálvez-López and Tardós, 2012), FABMAP (Cummins and Newman, 2008) and VLAD (Jégou et al., 2010). These are state of the art place recognition algorithms used by a number of SLAM systems for loop closure, and a previous work (Lowry and Andreasson, 2018) compared their performance on datasets that included SFU. We build on this comparison with new data and a new approach here.

Algorithmic approaches are compared against a deep convolutional neural network, NetVLAD, which consists of a layer approximating VLAD features built on top of any network trained to output place descriptors. As the NetVLAD authors note, the performance of their network is dependent on the descriptive capabilities of the underlying network, and so we include a primary investigation into one of these, VGG-16, trained to classify rather than match places, to confirm whether it has learned an appropriate level of descriptive capability for forest scenes. This incremental approach to evaluation allows us to build a case for NetVLAD's performance without the ability to compare to a ground truth.

We find the NetVLAD performs favourably even on difficult forest scenes, and so our final contribution is an offline integration as loop closure detection for ORBSLAM. While this appears to be successful in terms of NetVLAD proposing loop closures at appropriate points, ORBSLAM proves unable to resolve features between the matching views and so remains unable to close the loop.

8.1.1 Main Assumptions

As mentioned in the previous section, the first part of the work in this chapter is done under the assumption that NetVLAD's recognition performance is influenced by the classification performance of the network it is built on top of. This is why we confirm VGG-16's forest classification capability before moving on to the full NetVLAD experiments.

While place recognition and loop closure can be technically considered the same task, the practicalities of how they are deployed affect how they should be evaluated. Places are ill-defined for the continuous video data of SLAM problems and we cannot draw a line at a particular frame to declare it the start of end of a place. Likewise in the twisting sight lines of a relatively small scale forest track, we cannot be sure that parts of a previous place won't briefly come into view from elsewhere. Therefore, when we evaluate performance on this continuous data we assume that it is more helpful to look

for regions of similarity, instead of classifying True/False Positives/Negatives.

8.2 Publication: Lost in the Woods? Place Recognition for Navigation in Difficult Forest Environments

This work has been published in a special issue of Frontiers in Robotics and AI entitled Robotics Perception in Adversarial Environments.

This is all a personal contribution, with the exception of the guidance and editorial of written content provided by Prof Barbara Webb.

frontiers in Robotics and AI

ORIGINAL RESEARCH
published: 27 November 2020
doi: 10.3389/frobt.2020.541770

Lost in the Woods? Place Recognition for Navigation in Difficult Forest Environments

James Garforth and Barbara Webb*

School of Informatics, University of Edinburgh, Edinburgh, United Kingdom

Forests present one of the most challenging environments for computer vision due to traits, such as complex texture, rapidly changing lighting, and high dynamicity. Loop closure by place recognition is a crucial part of successfully deploying robotic systems to map forests for the purpose of automating conservation. Modern CNN-based place recognition systems like NetVLAD have reported promising results, but the datasets used to train and test them are primarily of urban scenes. In this paper, we investigate how well NetVLAD generalizes to forest environments and find that it out performs state of the art loop closure approaches. Finally, integrating NetVLAD with ORBSLAM2 and evaluating on a novel forest data set, we find that, although suitable locations for loop closure can be identified, the SLAM system is unable to resolve matched places with feature correspondences. We discuss additional considerations to be addressed in future to deal with this challenging problem.

Keywords: visual perception, place recognition, forests, scene statistics, navigation, SLAM, field robotics

OPEN ACCESS

Edited by:
Max Pfingsthorn,
Institute for Information Technology (OFFIS), Germany

Reviewed by:
Yunchao Tang,
Zhongkai University of Agriculture and Engineering, China
Johannes Pellenz,
BAAINBw, Germany

****Correspondence:***
James Garforth

Specialty section:
This article was submitted to Robot and Machine Vision, a section of the journal Frontiers in Robotics and AI

Received: *10 March 2020*
Accepted: *06 November 2020*
Published: *27 November 2020*

Citation:
Garforth J and Webb B (2020) Lost in the Woods? Place Recognition for Navigation in Difficult Forest Environments. Front. Robot. AI 7:541770. doi: 10.3389/frobt.2020.541770

1. INTRODUCTION

Mobile robotic systems have the potential to aid in forest management by improving the efficiency, scale and accuracy of tree health data gathering. In order to do this, robots must be able to navigate autonomously around large GPS-denied forest environments where modern visual Simultaneous Localization and Mapping (SLAM) techniques often struggle (Garforth and Webb, 2019).

An important part of the SLAM pipeline is loop closure, wherein the system recognizes that it has returned to a previously visited location (or "place") and can update its beliefs about intervening states, reducing the effect of drifting sensor readings. The place recognition algorithms used for loop closure in many classic SLAM techniques work by converting images to local descriptors and performing a search over previous descriptors for a likely match.

Deep learning approaches, particularly Convolutional Neural Networks (CNNs), such as NetVLAD (Arandjelovic et al., 2016), have demonstrated substantial promise for matching of images or places from databases. Less has been done to investigate their effectiveness at working for robotic loop closure applications. These often require fine place granularity, real-time performance, and the ability to deal with distinctive environments that are not classically represented in the large place datasets, such as forests.

In this paper, we identify NetVLAD as a candidate for improving loop closure in forest environments, and demonstrate its superior performance, compared against a baseline of traditional approaches, in a series of experiments. Our main contributions are as follows:

- Demonstrating the capacity of NetVLAD's underlying architecture for describing forest scenes.
- Comparison with state of the art loop closure techniques to demonstrate state of the art performance on challenging forest datasets, even in the presence of changing environmental conditions.
- Evaluation of NetVLAD as loop closure mechanism for a robot performing SLAM in a forest using a new data set.

2. RELATED WORK

2.1. Visual SLAM

Visual Simultaneous Localization and Mapping systems are typically feature-based, extracting salient features from video frames in order to track the motion of the camera over time. Since Klein and Murray (2007) this tracking tends to be done in its own thread to improve performance, separated from the less time critical task of building a 3D map of the world by feature triangulation. The influential ORBSLAM improves tracking performance further with fast ORB features, as well as multiple techniques for map optimization.

Other sensors than a single camera have been used for SLAM applications, such as stereo pairs or depth cameras (RGBD, as in Whelan et al., 2015). Both of these provide useful depth information to a SLAM system, but at the cost of power, weight, computational complexity and their own unique challenges. The setup of a stereo camera's baseline determines the range at which it can reliably estimate depth, and it must still be able to match features between its two images. An RGBD camera relies on reflected infrared light for depth estimation, and so is sensitive to ambient lighting or reflective properties of a scene. These problems are less prevalent under controlled laboratory conditions, but in the field many researchers still opt for a monocular camera setup.

Combining camera with inertial data, as in OKVIS (Leutenegger et al., 2015) and VINS-Mono (Qin et al., 2018), helps to improve local motion estimation, but requires carefully co-calibrated sensor data. This data is absent from most existing datasets, limiting a thorough analysis of visual-inertial SLAM performance.

Many recent works in monocular SLAM have revolved around replacing discrete features with aligning images directly through minimizing photometric error. The most successful of these are LSD-SLAM (Engel et al., 2014) and its successor DSO (Engel et al., 2017). These "direct" approaches have the advantage of not needing to detect corners in images and are therefore more robust to scenes containing large featureless regions like walls. Feature extraction also often accounts for much of the processing in feature-based SLAM, so direct systems can achieve performance benefits by forgoing it. These systems still have their limitations, however, and in their comparison of the two approaches, Yang et al. (2018) note that direct methods are more vulnerable to the effect of rolling shutter cameras and to highly dynamic lighting. The abundance of these in available forest data, as well as the lack of necessary photometric calibration, make feature-based methods an easier choice for mapping forests. Supporting the continued use of the feature-based approach, in Gao et al. (2018) the loop-closure enabled version of DSO is compared against ORBSLAM2 and accuracy of the two systems is notably similar.

2.2. Navigating and Mapping Forests

Traditionally, work in the field of forest mapping has been done from stationary or ground-based platforms, using heavy and power-hungry laser-based scanning, and requiring offline processing of the data to form a map (Takashi et al., 2014; Pierzchała et al., 2018). Mobile, online systems often extract only tree trunks from the scene, such as in Schultz et al. (2016) or Liao et al. (2016), a simplification that has also been used for RGBD systems in Fan et al. (2020). This avoids much of the complexity of the forest scene but also limits the usefulness of the final map to only information about placement of trees. While recent works show impressive results for improving laser-based mapping of the whole scene (Tinchev et al., 2018), laser-based sensors are still expensive and need a lot of power. A niche remains for navigating in real time using only monocular cameras, with the aim of enabling deployment on light, nimble and low cost Unmanned Aerial Vehicles (UAVs).

A number of recent works have looked at the problem of visual trail following under forest canopy, starting with Giusti et al. (2015) and more recently Maciel-Pearson et al. (2018). These represent only reactive navigation, rather than mapping, but what they do show us is the potential of neural network based approaches for cutting through the visual complexity of forest scenes. Silva et al. (2020) have recently improved such a trail following system by integrating depth information extracted from a SLAM system, LSD-SLAM (Engel et al., 2014), to augment their trail following, but do not evaluate its performance.

Natural environments like forests present a series of problems for traditional visual SLAM systems, as outlined in Garforth and Webb (2019). Firstly, they undergo frequent rapid changes in illumination due to the effect of the canopy blocking parts of the bright sky. Secondly, they contain a lot of in-scene motion as leaves and ground vegetation move with the wind. Finally, all of those plants result in an abundance of texture that can overwhelm feature detection algorithms.

2.3. Loop Closure

Visual SLAM systems keep track of the motion of the camera through the world over time, but due to sensor noise and other sources of inaccuracy, this ego-motion estimate experiences some drift. If the system can detect when it has returned to a previously visited location then it can estimate and optimize out drift that accrued in between. This is the purpose of loop closure. While loop closure and place recognition are similar tasks, they have notably different priorities and deployment or evaluation challenges. The "places" in place recognition datasets are sampled from a wide variety of geographic locations, can represent hundreds of square meters of space (e.g., a large city intersection) and do not usually overlap with each other. A proposed match here is clearly correct or incorrect. In a robotic loop closure context, input images are a continuous, if down-sampled, video stream with no discrete places and plenty of overlap. The challenge here is not to always find all of the closest matches for all images, but rather to recognize the general

similarity of the current location to a previously visited region, and thus initiate a process of view alignment.

Three state of the art loop closure algorithms are compared in Lowry and Andreasson (2018): a binary Bag of Words (BoW) model similar to that used in ORBSLAM2 (Gálvez-López and Tardós, 2012; Mur-Artal and Tardós, 2017), FAB-MAP (Cummins and Newman, 2008), and a Vectors of Locally Aggregated Descriptors (VLAD, Jégou et al., 2010). All of these methods reduce images to easily comparable descriptors that can be used to judge image similarity. Lowry and Andreasson compare performance of various descriptor sizes, on a number of datasets including one section of forest, and find that VLAD performs best in most cases.

2.4. Deep Learning and Place Recognition

Image matching is a popular domain for Deep Neural Networks, and when the images being matched against represent locations the task is known as place recognition (Lowry et al., 2015). In many areas of computer vision, DNNs have led to rapid advancement in image processing performance, demonstrating robustness to appearance (Gomez-Ojeda et al., 2015), lighting (Gomez-Ojeda et al., 2018), and viewpoint changes (Chen et al., 2017).

In the case of Kendall et al. (2015), Walch et al. (2017), and more recently Li et al. (2019), the output can be as complex as a 6 degree of freedom pose regression respective to the original "map" of images. What all of these systems have in common, however, is that the domain against which they can match novel images is fixed at the point at which they are trained. This makes them unsuitable for deployment in many real world robotic applications, such as loop closure, where the environment has not been previously mapped.

More suitable are networks designed to produce descriptors, similarly to classical approaches. Notable among these is NetVLAD (Arandjelovic et al., 2016), which is trained end-to-end for the place recognition task rather than using representations from a similar task, such as object classification. NetVLAD combines a layer replicating the descriptive power of VLAD descriptors on top of features learned from training on datasets of millions of scene images. The result is a network trained to produce feature vectors where the euclidean distance between two compared vectors should be smaller when the images they were produced from observe the same scene. NetVLAD's descriptors are small enough to be plausibly used for real time place matching, and have been shown to be robust to a lot of visual appearance change.

The CNN-based method of Chen et al. (2017) takes a different approach from whole-image descriptors, instead describing regions within images in a way that makes them robust to partial views of scenes. Vysotska and Stachniss (2019) have taken what could be considered the opposite approach and attempt to match not single image descriptors but sequences of image descriptors. In this way they adaptively define what a "place" is in each context, and are much better prepared to work with recognition in continuous video data. Unfortunately, this work does not have publicly available code, so we focus on the network they built upon: NetVLAD.

3. METHODS

3.1. Datasets

The forest datasets used in this work were mostly taken from Garforth and Webb (2019), and are summarized in **Figure 1**.

The SFU Mountain dataset (Bruce et al., 2015) follows the same route along a mountain path on 4 separate occasions under different environmental conditions (dry, wet, dusk, and night) and provides hand-identified matches between a sub sample of places along that route. We split it into "Road" and "Forest" sections based on when the vehicle turns onto a path under forest canopy.

The Hillwood datasets were recorded from two different aerial vehicles: Parrot's AR and Parrot Bebop drones. These represent more difficult forest paths with less of a defined track and less constrained in motion than the ground vehicle in SFU. We supplement these with a new dataset, Forest Loop, also recorded on a Bebop drone. The aim with this dataset was to follow a simple circuit and keep the motion as smooth as possible. In this way it is ideal for testing SLAM's frame to frame tracking as well as loop closure.

Finally, the Unreal dataset follows UAV-like paths similar to Hillwood and Forest Loop, but does so in a photorealistic simulated forest rendered with the Unreal game engine.

3.2. Recognizing Forests

Of the versions of NetVLAD reported in Arandjelovic et al. (2016), the most successful builds on VGG-16 (Simonyan and Zisserman, 2014). Before we test NetVLAD on the forest places task, we want to confirm that the underlying network has the ability to classify our datasets as forests. It is the descriptive ability of this network that NetVLAD's final pooling layer builds upon, so without it further tests would be unnecessary.

We use a Keras implementation of VGG-16 by Kalliatakis (2017) to evaluate whether the network can correctly identify forest data. The weights provided for this version of VGG-16 are pre-trained for classification on Places365 (Zhou et al., 2017), which according to the authors provides a "quasi-exhaustive" dataset of possible scene categories including a variety of different types of forest.

Each video frame is resized to a standard input size (224 × 224) and passed through the network, which provides an ordered list of possible scene category labels. The top prediction is used as the classification label for the frame. We record the percentage of frames classified as one of the "forest" labels, as well as noting other commonly assigned labels. This process is repeated for each dataset.

3.3. Recognizing Places in Forests

3.3.1. Setup

We set up NetVLAD using the Tensorflow code and pre-trained weights provided by Cieslewski et al. (2018). The only parameter that we have to set is how many of the principal descriptor dimensions to use in our descriptors. As the original authors establish a good trade-off between accuracy and size at 128 dimensions, and we are interested in real time performance, we use that value.

Name	SFU Forest	SFU Road	AR	Bebop	Forest Loop	Unreal
Count	4	4	3	1	2	5
FPS	30	30	12	30	30	30
Res	752x480	752x480	320x240	1920x1080	856 x 480	640x480
Eg.						

FIGURE 1 | Table showing datasets used in this work, with video count, frame rate, resolution and an example frame.

3.3.2. Evaluation

We duplicate the methodology described in Lowry and Andreasson (2018) for evaluating place recognition performance. NetVLAD is used to produce descriptors for each image in the database and each query image. For each query, the squared euclidean distance is calculated between the query descriptor and each of the database descriptors and the top N are considered as potential matches. If one of these matches is within D images of the correct match in the sequence then it is considered to be a true positive. As the datasets tested here are of a small scale compared to city-sized datasets NetVLAD was originally demonstrated on, values of N and D are set to 1 and 2, respectively.

To make this task more difficult, database and query images are also cropped from the full image frame such that they are offset by 40% from one another.

3.3.3. Evaluating on Data With Ground Truth

We use the same two SFU video pairs as Lowry and Andreasson (2018): with the dry conditions as our database set and the wet or dusk conditions as query sets. For fair comparison with the original results we report performance for the whole datasets, but as we are specifically interested in performance under the more challenging canopy covered section, we also report results for a test set with the open road sections removed. Note that in this case the database set is not reduced, as we would expect this to artificially improve performance.

3.3.4. Evaluation on Data With No Ground Truth

We wish to evaluate performance on more forest data than just the SFU videos, but what little data is available does not have ground truth, primarily due to the poor performance of GPS under canopy cover. Instead we use a methodology of splitting a single video into two by alternately sampling frames. This way, our database and query videos do not contain any of the same frames but two frames sampled from next to each other should usually still represent the same place.

There are two parameters to consider when using this method. First, the step size between places in a video, which we set to 3 s (36 or 90 frames for Hillwood AR and Bebop, respectively) in order to generate an equivalent number of places as in the SFU dataset (∼150). Given the speed of the UAV recording the Hillwood data, this offset should represent a maximum of 5 m between places. The second parameter is the offset between the database and query frames. At an offset of 0, the sets would be identical, whereas if the offset equals the step size the "ground truth" identity of the frames would be misaligned. We test a range of values between these two unhelpful extremes.

3.4. Loop Closure

Our final evaluation is of NetVLAD's potential as a loop closure mechanism in visual SLAM, which we perform in two ways. Firstly, we compare the descriptors generated for each dataset route against themselves and plot the distances as a confusion matrix to look for regions of low distance at the sites of known loop closures. Settings for distance thresholds or other mechanisms of identifying a loop closure will vary between SLAM systems, but usually rely on repeated recognition of similarity within a short period of time. This method allows us to evaluate whether regions of similarity emerge from the system without artificially defining the boundaries of "places" within the data.

Secondly. we perform an integration with ORBSLAM2 (Mur-Artal et al., 2015), a popular visual SLAM system, which uses DBoW2 (Gálvez-López and Tardós, 2012), a Bag of Words method, to propose potential loop closures. We replaced DBoW2 with a simple threshold on the distance between NetVLAD descriptors to decide when to propose an image as a loop closure. Note at this stage the test system is not optimized for running in real time and so the image descriptors are pre-calculated for the whole dataset, and loaded from a file for comparison at runtime. ORBSLAM2 evaluates our loop closures as usual, extracting features from new and old images to try and align them.

This system needed a dataset to be tested on, as previous work (Garforth and Webb, 2019) has shown that SLAM has difficulty tracking forest videos, and in order to test a loop closure we need continuous tracking between the first and second visit to the loop location. This has not been achieved on Hillwood, and SFU contains only challenging 180° rotated loop closures. Our new "Forest Loop" dataset was gathered specifically to provide both a smooth motion to aid in tracking and a closed circuit to test our loop closure system on real robot data.

4. RESULTS

4.1. Classifying Forests

As can be seen in **Table 1**, VGG-16 achieves reasonable classification of forests on all real datasets. More than half of the misclassifications of SFU Forest dry are as "field road" which is reasonable given that some open track still exists in the dataset. Hillwood AR is primarily misclassified as "trench," "driveway," or

TABLE 1 | Percentage of correct classification matches achieved by NetVLAD when using a different camera from the stereo pair for the query vs. database dataset.

Dataset	Forest	Other common labels
Hillwood Bebop	99.10%	Orchard (0.28%), trench (0.20%)
SFU Forest Dry	86.31%	Field road (9.64%), corn field (1.18%)
Hillwood AR	80.00%	Trench (6.21%), driveway (3.41%), yard (2.51%)
Forest Loop	77.69%	Trench (14.81%), yard (2.21%), landfill (1.66%)
Unreal	60.95%	Fishpond (15.84%), pond (5.71%), aquarium (1.87%)

TABLE 2 | Percentage of correct place matches achieved by different place recognition algorithms on pairs of routes under different environmental conditions from the SFU dataset.

Datasets	BOW	FAB-MAP	VLAD	NetVLAD Full	NetVLAD Forest
Dry-Dry	–	–	–	97.3	93.9
Dry-Wet	12.6	18.4	28.5	37.0	50.0
Dry-Dusk	15.1	22.6	32.2	47.3	53.3
Dry-Night	–	–	–	9.59	30.0

Results for the first three algorithms are from the work of Lowry and Andreasson (2018). The fourth column reports on NetVLAD, and the final covers the same algorithm but reports only for the final canopy-covered section of the dataset.

"yard." The reason for these is less obvious, but this dataset was recorded at a time of year when a lot of leaves were on the ground, a feature that was perhaps also present in some of the original training data for "driveway" and "yard."

The simulated forest data is classified significantly less accurately than real data, with the most common non-forest labels being pond, fishpond and aquarium (which account for 23.63% of classifications between them). The artificial lighting conditions seem the most likely explanation for this, which is not unsurprising given the results of Garforth and Webb (2019) and furthers their warning about using simulated data to test algorithms meant to work in the real world.

Aside from in simulation, initial results show that the network underlying NetVLAD is able to learn features specific to forests, which reassures us that full NetVLAD has the potential for describing places within forests.

4.2. Place Recognition Against Baseline

Table 2 shows the rates of correct matches achieved by NetVLAD between pairs of routes from the SFU dataset under various environmental conditions. These are presented alongside the results for other place recognition systems from Lowry and Andreasson (2018) for the same pairings, where available.

The rates of correct matches in **Table 2** show NetVLAD achieves a substantial increase over VLAD features, as well as BoW and FAB-MAP, in both previously tested forest setups. Notably we have taken the best result across descriptor sizes for BoW, FAB-MAP, and VLAD, so in most cases our 128 dimension NetVLAD descriptor would also use less memory. This makes a strong case for using it in the loop closure settings where those other descriptors are typically used.

TABLE 3 | Percentage of correct place matches when sub-sampling a single video to produce database and query sets.

		Offset (as fraction of step)				
Dataset	Step size	1/6	1/3	1/2	2/3	5/6
Hillwood AR	36	64.34	63.64	60.14	58.45	64.09
Hillwood Bebop	90	75.83	65.0	69.17	71.43	71.43

Step size (in frames) indicates rate at which frames are sampled and is set to the equivalent of 3 s. Offset indicates distance between frames assigned to different sets.

Somewhat surprisingly, NetVLAD also performs better on the canopy-covered sections of forest at the end of the SFU datasets than it does on the track as a whole. The track at the start of the SFU videos must be difficult to distinguish, likely due to most frames consisting solely of a road and surrounding treeline. To the eye these images are very similar, as noted in Garforth and Webb (2019). Under the canopy, however, the more complex skyline has been shown to be useful in navigation before (Stone et al., 2016), so we posit that this is part of what the network is using to achieve its improved performance.

Under canopy cover NetVLAD also shows resistance to changing environmental conditions, achieving performance on the difficult dry-night pairing that matches VLAD's performance on the two easier pairings. The skyline is not visible in the night time data, so the performance here implies that this isn't the only useful feature being learned.

4.3. Evaluation Without Ground Truth

Table 3 shows the match percentages when comparing two datasets produced by subsampling from a single video. We varied the number of frames offset between the two subvideos and as would be expected we find the performance to be better when this is closer to 0 or to the step size, as this means frames recorded closer in time exist in the two subvideos. Even at the worst case for either dataset (58.45% for Hillwood AR or 65.0% for Bebop) performance does not diminish too much, suggesting a resistance to translations in the NetVLAD descriptors.

4.4. Loop Closure

We generated confusion plots of distances between pairs of NetVLAD descriptions for videos from robots traversing forest terrain. We can make out columns of blue (low distance) in **Figures 2**, **3** at the indicated points where the path loops back to previous visited locations, showing that the descriptor comparison is able to successfully detect the similarity of these forest locations. There are some other patches at points we had not identified as full loops, but it is worth noting that in forested conditions especially these "places" are not straightforward to define, and parts of one scene may indeed be visible in a part of another view from another part of the route, so some noise is expected. Occasional incorrect matches would be dealt with in a SLAM application through filtering and feature correspondence checks. Additionally, as a baseline check, we compared two different traversals of the same Forest Loop route in **Figure 4** and can see a clear path of high similarity near the diagonal between the two videos.

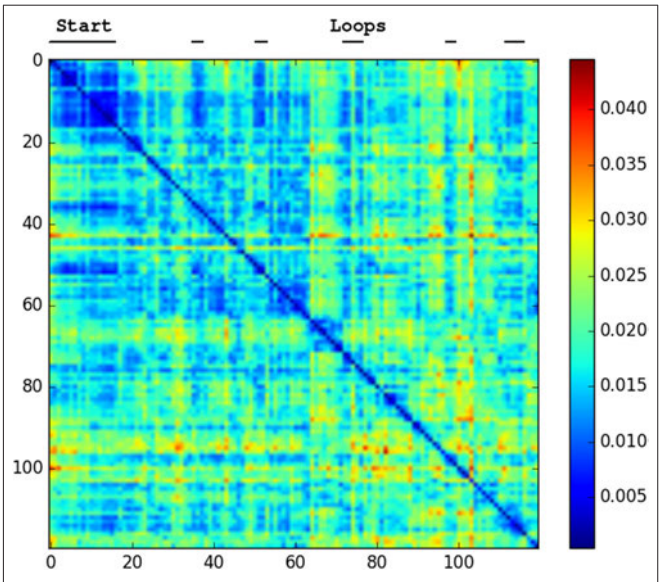


FIGURE 2 | Confusion matrix showing the squared euclidean distance between frames of Hillwood Bebop. Note multiple loops returning to the start as shorter distance patches.

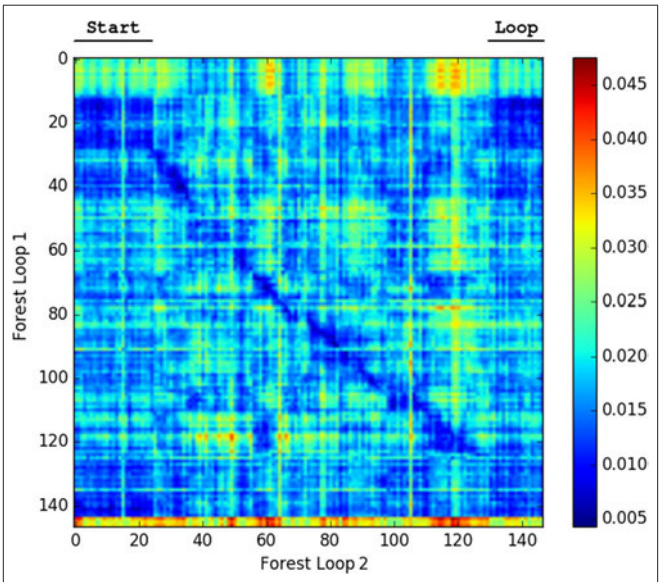


FIGURE 4 | Confusion matrix showing the squared euclidean distance between frames of Forest Loop on two different passes. The matching of the route near the diagonal and the loop are both visible.

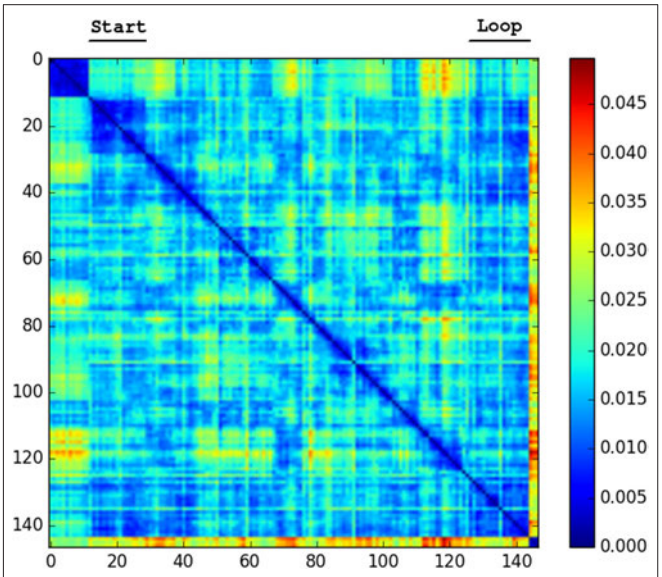


FIGURE 3 | Confusion matrix showing the squared euclidean distance between frames of Forest Loop. Loop closure at the end of the video noticeable as shorter distance patch.

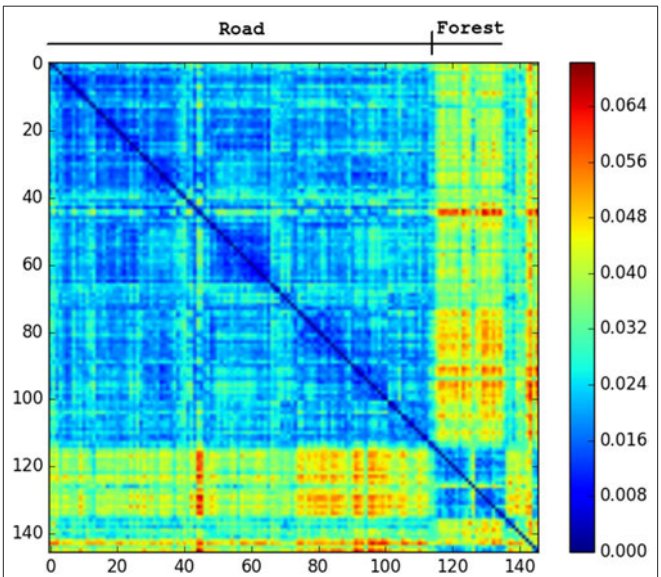


FIGURE 5 | Confusion matrix showing the squared euclidean distance between frames of SFU dry. Road and forest sections very different from each other. No loop matches.

The only repeated traversals of the same path in the SFU video occur while facing in opposite directions. These are challenging conditions for any place recognition, so this video is test of how far NetVLAD can be pushed. **Figure 5** shows that the descriptor comparison makes a clear distinction between forest and road sections of the SFU video, but within these sections most places seem equally similar to each other. We would not expect to be able to close the loop in this dataset with NetVLAD.

If we look at sample frames from the regions NetVLAD indicates are similar in **Figure 6**, we can see that the network is producing low distance scores in spite of the presence of potentially confusing factors, such as a rotated camera view, a prominent object (tree) not previously visible, and a change in lighting affecting the apparent texture of the ground.

The results so far demonstrate this network's potential for use as a loop closure mechanism for SLAM in forests.

FIGURE 6 | Place matches achieved on Hillwood Bebop dataset demonstrate robustness to rotation [view is shifting right from **(A)** to **(D)**], illumination changes [sun passes behind cloud between **(A)** and **(B)**] and onset of previously unseen occluding objects [large tree comes into view in foreground between **(C)** and **(D)**].

FIGURE 7 | Example correct place match proposed by NetVLAD between the beginning **(A)** and end **(B)** of Forest Loop.

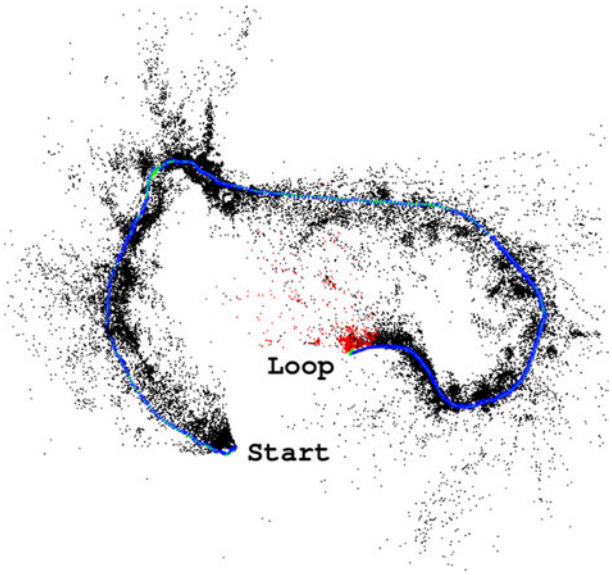


FIGURE 8 | ORBSLAM2 (Mur-Artal et al., 2015) successfully tracks the Forest Loop video, but inevitable drift results in the start and end locations not connecting as they should.

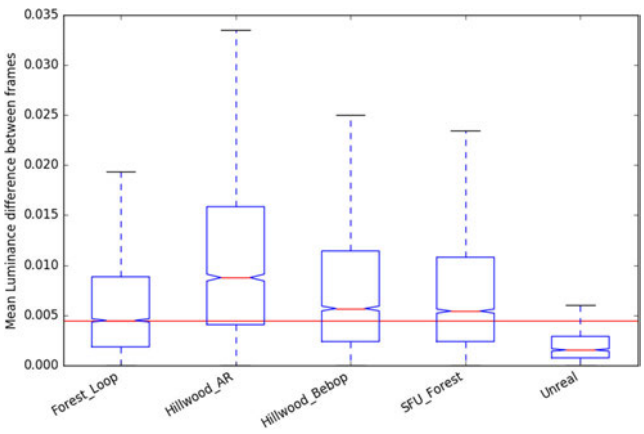


FIGURE 9 | Luminance change statistics of the new "Forest Loop" dataset, compared to existing.

Given this, our final experiment was the integration of NetVLAD descriptors in ORBSLAM2 as a loop closure proposal mechanism. Unfortunately, even though the system proposed loop closures at appropriate frames (such as those shown in **Figure 7**), the alignment of the point clouds required for the global map optimization always failed to align the features from the two matched frames. The resulting map can be seen in **Figure 8** with the loop still open.

The likely cause of these difficulties is the very highly textured scenes providing an abundance of potential features, or the notable change in lighting that occurs even in the few minutes between the images in **Figure 7**, such that repeated visits to the same location do not have a large enough overlap in extracted features to find an alignment. This was a risk, given the previously reported poor performance of SLAM systems in these environments in Garforth and Webb (2019), and shows that more work must be done on the frame to frame tracking part of the mapping process before a full working forest SLAM system can be produced.

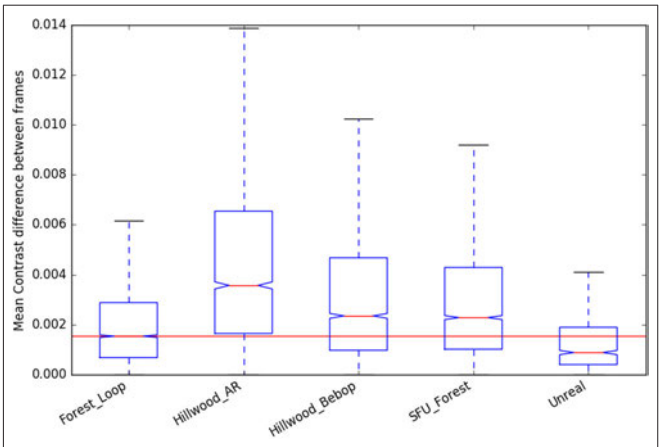


FIGURE 10 | Contrast change statistics of the new "Forest Loop" dataset, compared to existing.

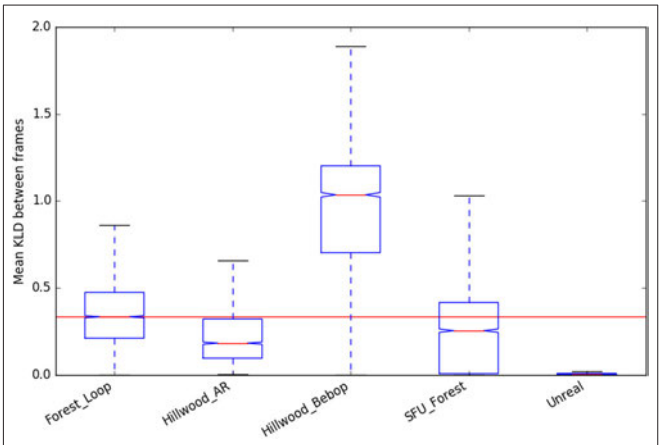


FIGURE 11 | KL Divergence statistics (used as a representation of in-scene motion) of the new "Forest Loop" dataset, compared to existing.

4.5. Scene Statistics

As we are reporting work with a new forest dataset, we present here the scene statistics (as described in Garforth and Webb, 2019) of that data, compared to the previously existing datasets. This data was also recorded using the Bebop Drone, so it notably has the same camera properties as "Hillwood_Bebop."

We can see that "Forest_Loop" is less variable in its lighting (**Figures 9**, **10**) than other forest datasets, while its in scene motion (as measured by KL divergence in **Figure 11**) is indistinguishable. No specific care was taken to avoid areas based on lighting, but this data was recorded at a careful pace, keeping a good distance away from foliage in order to improve matching chances. Easier to track data forest data was our aim, and these statistics would lead us to believe it has been at least partially achieved.

5. CONCLUSION AND FUTURE WORK

In this paper we presented an investigation into the use of place recognition network NetVLAD for loop closure in challenging forest mapping scenarios. Even though NetVLAD itself has not been trained to recognize places in forests, we showed that its underlying VGG-16 network does categorize a variety forest datasets correctly. We then demonstrated that NetVLAD performs better than state of the art loop closure approaches in this difficult environment, robust to the presence of changing lighting, time of day, weather conditions, rotation, and translation. Our results marked NetVLAD as a candidate for proposing loop closures for SLAM, and we put together a test system as well as gathering new data to confirm. We found that NetVLAD reliably proposed loop closures to ORBSLAM, but that the frame to frame ORB feature matching of the SLAM system was not able to integrate these into the map. This is a useful warning for anyone attempting SLAM in forests, and supports feature matching as one avenue for further research.

A number of other improvements can be made to our test integration with SLAM to produce a system ready for deployment. Given ORBSLAM's difficulty aligning feature points at proposed loop closures, we would propose investigating a feature-less "direct" SLAM method like DSO instead. Once a pairing with good performance is found, we would replace our offline NetVLAD descriptor calculation with one that runs in real-time. Finally, to improve scalability as the map gets larger we would consider a hierarchical graph and sequence based search method, like that described in Vysotska and Stachniss (2019).

DATA AVAILABILITY STATEMENT

The SFU dataset can be found at http://autonomy.cs.sfu.ca/sfu-mountain-dataset/. The rest of the data supporting the conclusions of this article will be made available by the authors, without undue reservation, to any qualified researcher.

AUTHOR CONTRIBUTIONS

The work for this paper was undertaken by JG. Supervision and proof reading were provided by BW. All authors contributed to the article and approved the submitted version.

FUNDING

This work was supported by the Edinburgh Centre for Robotics and the Engineering and Physical Sciences Research Council.

REFERENCES

Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., and Sivic, J. (2016). "NetVLAD: CNN architecture for weakly supervised place recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Las Vegas, NV), 5297–5307. doi: 10.1109/CVPR.2016.572

Bruce, J., Wawerla, J., and Vaughan, R. (2015). "The SFU mountain dataset: semi-structured woodland trails under changing environmental conditions," in *IEEE International Conference on Robotics and Automation 2015, Workshop on Visual Place Recognition in Changing Environments* (Seattle).

Chen, Z., Maffra, F., Sa, I., and Chli, M. (2017). "Only look once, mining distinctive landmarks from convnet for visual place recognition," in *2017 IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS) (Vancouver, BC: IEEE), 9–16. doi: 10.1109/IROS.2017.8202131

Cieslewski, T., Choudhary, S., and Scaramuzza, D. (2018). "Data-efficient decentralized visual SLAM," in *2018 IEEE International Conference on Robotics and Automation (ICRA)* (Brisbane, QLD: IEEE), 2466–2473. doi: 10.1109/ICRA.2018.8461155

Cummins, M., and Newman, P. (2008). Fab-map: probabilistic localization and mapping in the space of appearance. *Int. J. Robot. Res.* 27, 647–665. doi: 10.1177/0278364908090961

Engel, J., Koltun, V., and Cremers, D. (2017). Direct sparse odometry. *IEEE Trans. Pattern Anal. Mach. Intell.* 40, 611–625. doi: 10.1109/TPAMI.2017.2658577

Engel, J., Schöps, T., and Cremers, D. (2014). "LSD-SLAM: large-scale direct monocular SLAM," in *Computer Vision–ECCV 2014, Volume 8690 of Lecture Notes in Computer Science*, eds D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars (Cham: Springer International Publishing), 834–849. doi: 10.1007/978-3-319-10605-2_54

Fan, Y., Feng, Z., Shen, C., Khan, T. U., Mannan, A., Gao, X., et al. (2020). A trunk-based SLAM backend for smartphones with online SLAM in large-scale forest inventories. *ISPRS J. Photogramm. Remo. Sens.* 162, 41–49. doi: 10.1016/j.isprsjprs.2020.02.006

Gálvez-López, D., and Tardós, J. D. (2012). Bags of binary words for fast place recognition in image sequences. *IEEE Trans. Robot.* 28, 1188–1197. doi: 10.1109/TRO.2012.2197158

Gao, X., Wang, R., Demmel, N., and Cremers, D. (2018). "LDSO: direct sparse odometry with loop closure," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Madrid: IEEE), 2198–2204. doi: 10.1109/IROS.2018.8593376

Garforth, J., and Webb, B. (2019). "Visual appearance analysis of forest scenes for monocular SLAM," in *2019 International Conference on Robotics and Automation (ICRA)* (Montreal, QC: IEEE), 1794–1800. doi: 10.1109/ICRA.2019.8793771

Giusti, A., Guzzi, J., Cireşan, D. C., He, F.-L., Rodríguez, J. P., Fontana, F., et al. (2015). A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robot. Autom. Lett.* 1, 661–667. doi: 10.1109/LRA.2015.2509024

Gomez-Ojeda, R., Lopez-Antequera, M., Petkov, N., and Gonzalez-Jimenez, J. (2015). Training a convolutional neural network for appearance-invariant place recognition. *arXiv* 1505.07428.

Gomez-Ojeda, R., Zhang, Z., Gonzalez-Jimenez, J., and Scaramuzza, D. (2018). "Learning-based image enhancement for visual odometry in challenging HDR environments," in *2018 IEEE International Conference on Robotics and Automation (ICRA)* (Brisbane, QLD: IEEE), 805–811. doi: 10.1109/ICRA.2018.8462876

Jégou, H., Douze, M., Schmid, C., and Pérez, P. (2010). "Aggregating local descriptors into a compact image representation," in *CVPR 2010–23rd IEEE Conference on Computer Vision & Pattern Recognition* (IEEE Computer Society), 3304–3311. doi: 10.1109/CVPR.2010.5540039

Kalliatakis, G. (2017). *Keras-vgg16-places365*. Available online at: https://github.com/GKalliatakis/Keras-VGG16-places365

Kendall, A., Grimes, M., and Cipolla, R. (2015). "Posenet: a convolutional network for real-time 6-DOF camera relocalization," in *Proceedings of the IEEE International Conference on Computer Vision* (Santiago), 2938–2946. doi: 10.1109/ICCV.2015.336

Klein, G., and Murray, D. (2007). "Parallel tracking and mapping for small AR workspaces," in *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality* (Nara: IEEE), 1–10. doi: 10.1109/ISMAR.2007.4538852

Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., and Furgale, P. (2015). Keyframe-based visual-inertial odometry using nonlinear optimization. *Int. J. Robot. Res.* 34, 314–334. doi: 10.1177/0278364914554813

Li, X., Wang, S., Zhao, Y., Verbeek, J., and Kannala, J. (2019). Hierarchical scene coordinate classification and regression for visual localization. *arXiv* 1909.06216. doi: 10.1109/CVPR42600.2020.01200

Liao, F., Lai, S., Hu, Y., Cui, J., Wang, J. L., Teo, R., et al. (2016). "3D motion planning for UAVs in GPS-denied unknown forest environment," in *2016 IEEE Intelligent Vehicles Symposium (IV)* (Gothenburg: IEEE), 246–251. doi: 10.1109/IVS.2016.7535393

Lowry, S., and Andreasson, H. (2018). Lightweight, viewpoint-invariant visual place recognition in changing environments. *IEEE Robot. Autom. Lett.* 3, 957–964. doi: 10.1109/LRA.2018.2793308

Lowry, S., Sünderhauf, N., Newman, P., Leonard, J. J., Cox, D., Corke, P., et al. (2015). Visual place recognition: a survey. *IEEE Trans. Robot.* 32, 1–19. doi: 10.1109/TRO.2015.2496823

Maciel-Pearson, B. G., Carbonneau, P., and Breckon, T. P. (2018). "Extending deep neural network trail navigation for unmanned aerial vehicle operation within the forest canopy," in *TAROS* (Bristol). doi: 10.1007/978-3-319-96728-8_13

Mur-Artal, R., Montiel, J., and Tardos, J. (2015). ORB-SLAM: a versatile and accurate monocular SLAM system. *arXiv* 1502.00956. doi: 10.1109/TRO.2015.2463671

Mur-Artal, R., and Tardós, J. D. (2017). ORB-SLAM2: an open-source slam system for monocular, stereo, and RGB-D cameras. *IEEE Trans. Robot.* 33, 1255–1262. doi: 10.1109/TRO.2017.2705103

Pierzchała, M., Giguère, P., and Astrup, R. (2018). Mapping forests using an unmanned ground vehicle with 3D LIDAR and graph-SLAM. *Comput. Electron. Agric.* 145, 217–225. doi: 10.1016/j.compag.2017.12.034

Qin, T., Li, P., and Shen, S. (2018). Vins-mono: a robust and versatile monocular visual-inertial state estimator. *IEEE Trans. Robot.* 34, 1004–1020. doi: 10.1109/TRO.2018.2853729

Schultz, A., Gilabert, R., Bharadwaj, A., de Haag, M. U., and Zhu, Z. (2016). "A navigation and mapping method for UAs during under-the-canopy forest operations," in *2016 IEEE/ION Position, Location and Navigation Symposium (PLANS)* (IEEE), 739–746. doi: 10.1109/PLANS.2016.7479768

Silva, A., Mendonça, R., and Santana, P. (2020). Monocular trail detection and tracking aided by visual SLAM for small unmanned aerial vehicles. *J. Intell. Robot. Syst.* 97, 531–551. doi: 10.1007/s10846-019-01033-x

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

Stone, T., Differt, D., Milford, M., and Webb, B. (2016). "Skyline-based localisation for aggressively manoeuvring robots using UV sensors and spherical harmonics," in *2016 IEEE International Conference on Robotics and Automation (ICRA)* (Stockholm: IEEE), 5615–5622. doi: 10.1109/ICRA.2016.7487780

Takashi, T., Asuka, A., Toshihiko, M., Shuhei, K., Keiko, S., Mitsuhiro, M., et al. (2014). Forest 3D mapping and tree sizes measurement for forest management based on sensing technology for mobile robots. *Springer Tracts Adv. Robot.* 92, 357–368. doi: 10.1007/978-3-642-40686-7_24

Tinchev, G., Nobili, S., and Fallon, M. (2018). "Seeing the wood for the trees: reliable localization in urban and natural environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Madrid: IEEE), 8239–8246. doi: 10.1109/IROS.2018.8594042

Vysotska, O., and Stachniss, C. (2019). Effective visual place recognition using multi-sequence maps. *IEEE Robot. Autom. Lett.* 4, 1730–1736. doi: 10.1109/LRA.2019.2897160

Walch, F., Hazirbas, C., Leal-Taixe, L., Sattler, T., Hilsenbeck, S., and Cremers, D. (2017). "Image-based localization using lstms for structured feature correlation," in *Proceedings of the IEEE International Conference on Computer Vision* (Venice), 627–637. doi: 10.1109/ICCV.2017.75

Whelan, T., Leutenegger, S., Salas-Moreno, R., Glocker, B., and Davison, A. (2015). "ElasticFusion: dense SLAM without a pose graph," in *Robotics: Science and Systems* (Rome). doi: 10.15607/RSS.2015.XI.001

Yang, N., Wang, R., Gao, X., and Cremers, D. (2018). Challenges in monocular visual odometry: photometric calibration, motion bias, and rolling shutter effect. *IEEE Robot. Autom. Lett.* 3, 2878–2885. doi: 10.1109/LRA.2018.2846813

Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., and Torralba, A. (2017). Places: a 10 million image database for scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* 40, 1452–1464. doi: 10.1109/TPAMI.2017.2723009

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

8.3 Conclusions and Future Work

This paper presented an investigation into the use of NetVLAD for loop closure detection in forests. The case for NetVLAD's usefulness is built in stages. First, despite not being trained to recognise places in forests, NetVLAD's underlying VGG-16 network is shown to be able to classify them, indicating a useful descriptive capacity. NetVLAD is then shown to perform better than state of the art loop closure detection approaches, robust to the presence of changing conditions, at detecting places in several forest datasets. Finally, NetVLAD is integrated as loop closure detection for ORBSLAM. While it proposes correct loop locations, the feature matching of ORBSLAM remains unable to align the two views.

For the conclusions of this work to be applied in a real world system, it is necessary to solve the problem of feature matching between the two views that close a loop. One method for addressing this would be to be to forgo feature-based SLAM entirely and attempt integration with a "direct" method like DSO. These methods are known to have difficulties with variable lighting, however, and would be likely to have their own problems aligning the two views. The alternative is to work on improving the matching performance of ORBSLAM. Given the hypothesis that complex texture and lighting are the challenging traits of these scenes, and the knowledge that biological visual systems are adapted to deal with these traits, nature can be looked at for inspiring potential solutions.

8.3.1 Feature Correspondences

This work found that, even given an accurate loop closure proposal by NetVLAD, ORBSLAM would still fail to resolve features matches between the old and new view of the scene. This adds support to our analysis in the previous chapters than feature matching is a key area to improve upon in order to improve SLAM forest performance, and reinforces our attempt to address this in Chapter 7.

Chapter 9

Conclusion

This thesis has presented an investigation into the performance of visual SLAM in forests. The majority of SLAM research has been done in structured, man-made environments such as the insides of buildings or city streets, and this runs the risk of unintentionally biasing successful algorithms towards these kinds of scenes. Solutions are developed that have never been exposed to or evaluated on environments filled with high density texture, where large portions of the scene are in motion, or where lighting is always changing, nor is it well understood where or when these factors impact SLAM performance. Inevitably, combined with the publication of research code with minimal documentation or support, this leads to challenges in performing exhaustive testing, understanding the true limitations of individual systems, or deploying robots in important new application domains.

Development of SLAM for forest scenes is limited by the availability of evaluation data and techniques for this new domain. Track and map ground-truthing for forests does not exist and simulation, which might provide some replacement, proves to poorly reflect real forest data. Likewise, when evaluating place recognition for SLAM loop closure in an environment not divided up cleanly by walls or buildings, it is less straightforward to even define a “place”.

Much of this thesis work involved finding techniques and metrics by which mean-

ingful evaluation could be performed under challenging conditions, and developing the understanding of the problem domain rather than simply proposing a solution tailored to a overly specified use case. A suite of code has been produced allowing for combinations of dataset subsampling, statistical tests of data, and evaluations of performance with and without full SLAM systems, all to support the contributions of this work, along with new datasets and simulation environment. We intend to document and then make available both the code and datasets in the near future.

9.1 Contributions

The first contribution of this thesis was to compare performance of state of the art monocular visual SLAM systems on new and existing forest datasets, demonstrating that all of them struggle with this task and some do not manage to map at all. To understand this outcome, we proposed a suite of visual scene statistics to measure key traits in video data known to cause difficulties in navigation. The evaluation showed that forest data presents significantly more of these challenging traits. This work also presented a new simulated forest dataset that, while photorealistic to the human eye, still fails to reflect the statistical traits of the real forest. This serves as a warning against relying too heavily on simulation to develop navigation capabilities for new domains. The field of visual navigation in forests remains an active one with hard problems, to which this work has contributed insights. Indeed, the authors of the FinnForest dataset (Ali et al., 2020) cite our paper as motivation and guidance in creating a ground-truthed forest dataset. This work has also been cited by Maciel-Pearson et al. (2019) in exploring forests with a trail following MAV and by Tremblay et al. (2020) in their consideration of how to build a forestry automation system.

Our second contribution was an in-depth follow-up investigation into the factors effecting poor performance of SLAM systems in a forest setting, to determine whether some limitations of the data or implementation were influencing the result. We pro-

duced a new forest dataset with ideal initialisation conditions and careful camera motions. We found that while initialisation is more reliable on this dataset for two of our tested systems, their performance on the rest of the data is still unreliable, susceptible to loss of tracking at corners or in the presence of temporary camera artefacts, or frequently confused by the visual similarity between different forest locations. A set of experiments then explored the impact of changing each system's parameters on its performance. This showed potential initialisation performance improvements when the number of features extracted from each image is increased, either directly or by reducing the size of the image pyramid in ORBSLAM. All other changes from default parameters made little difference to performance or made it worse. Finally we delved into the code for each system, identifying the subcomponents involved at the points where failures seem to occur. We concluded that while some improvements can be made with careful motion of the platform or tuning of the parameters, the performance of monocular SLAM systems lacks adequate robustness to forests as an environment.

The third contribution presented an improved forest simulation with controlled variation of environmental conditions and used this to investigate the relationships between traits of the scene and our scene statistics, as well as between traits and monocular SLAM performance. While it proved a challenge to isolate traits such as lighting and scene motion from each other, even with the control granted by a simulated environment, this generally reflected correlation between these traits in the real world as well. For example, increased wind moving the canopy also increases the rate of change of lighting partially blocked by that canopy. We found the new simulation to more closely reflect real forest data in a number of ways, but that significant dissimilarity remains and real forest data is still needed to assess SLAM performance. Finally, experiments confirmed most expectations about the impact of environmental traits on SLAM systems; reductions in lighting changes in the Cloud conditions made initialisation easier, while extra in-scene motion caused by the Wind conditions made it more difficult. The Winter condition, reducing the complexity of texture in the scene, had

unexpectedly little impact, but has also revealed potential limitations of our simplified texture measure.

In the process of this thesis work, a modular computer vision system, EnVisionEd, has been developed which allows for rapid configuration of vision pipeline tests, both for evaluating statistics of datasets and for modifying input to SLAM systems. The flexibility of this system is the reason that it was used in every piece of work presented in this thesis, and the initial work put into its design paid off in speeding up later experiments. The fourth contribution took particular advantage of the rapid prototyping enabled by EnVisionEd to test the effects of a variety of preprocessing steps on feature matching in forest scenes and overall SLAM performance, quickly swapping out single processing components between a large number of tests while keeping the rest of the pipeline constant. Its major limitation is its inefficiency; in enabling modules to be agnostic to each other, the central Data object passed around between them can become home to redundant copies of data, and the decoupling between module's processes can prevent clever efficiency tricks that might be deployed in more optimised code. Speed was never an issue for any of our experiments, but any complex real time processing of video is not within the scope of the current version of our system. The re-usability of EnVisionEd has also already allowed two MSc projects to extend it, investigating new environments (underwater and surgical) and new scene statistics (noise, colour distribution and spatial correlation). Over sustained research use, often under time pressure, however, EnVisionEd's codebase has become disjointed, with older modules not being updated to reflect all of the framework capabilities added for later experiments. A third MSc student is now working on cleaning up the code, making sure the API is consistent and documented, and packaging it for wider use.

The final contribution was a publication assessing the performance of place recognition algorithms for SLAM loop closure in forest data. This compared state of the art convolutional neural architecture NetVLAD with popular algorithmic approaches (FABMAP, DBoW2 and VLAD) and found that both it and its underlying scene recog-

nition network generalise better to these new scenes. This solution was integrated with ORBSLAM as its loop closure detector, but even with correct loop closure matches proposed, ORBSLAM was unable to resolve feature matches between two views of the same scene.

9.2 Future Work

9.2.1 Preprocessing for Forests

While the preprocessing experiments in Chapter 7 had limited conclusions for improving overall feature matching performance, they do indicate both possible efficiency gains and that useful feature information is not evenly distributed across colour channels or image regions. Useful future work here would be to test these methods on a greater range of benchmarking datasets, such as our improved forest simulation, and to investigate combinations of these simple preprocessing methods. For example, it could be tested whether colour channel usefulness varies across image regions as suggested by the varying distributions of colour receptors across animal eyes. Evaluation of any promising preprocessing should further establish whether they improve the rate of loop closure for ORBSLAM on Forest Loop or LSDSLAM on Forest Loop 2.

Processing steps were also kept simple in this work, but there are many more ways to break down spectral sensitivity than just selecting channels from RGB, such as weighting a combination adaptively (as in Sun and Lau (2019)), or using colour opponency, and regional focus could be modelled as an attentional preference over image regions, rather than simple cropping, by using an attention model like GradCam (Selvaraju et al., 2017). It is possible that these more in-depth modellings of animal visual traits would have a greater effect on accuracy, but at the cost of more complex processing.

9.2.2 SLAM in Forests

The goal of producing a full forest-focused monocular SLAM system remains uncompleted in the time since our first publication, as reflected in the open research priorities set out by McGlade et al. (2022). We identified a range of existing tracking and loop closure detection components, and determined the best performing on existing and new forest datasets. Yet even combining and tuning these, performance is not of an appropriate level for robust deployment for forest mapping tasks. Here we discuss what seems to us the most promising approaching that might be explored to improve performance.

Feature detection might yet benefit from continuation of the preprocessing work as discussed above, but there are other possible approaches to this. ORB features could be replaced all together, and a possible starting point for this might be deep learnt features like LIFT (Yi et al., 2016), given how well NetVLAD was found to adapt to a new environment. Alternatively, new features might not be needed if the system can distinguish which parts of the scene are unreliable, by semantic segmentation of foliage or by identifying parts which are in motion as in works such as Scona et al. (2018). Features might be removed entirely by experimenting with direct SLAM systems instead, although these are more sensitive to lighting conditions, as evidenced by the poor performance of DSO (Engel et al., 2017) in our experiments, and this limitation would need addressing given the large lighting changes in forest data. Direct SLAM has the benefit of more dense reconstruction, and if the goal of measuring tree trunks is to be realised then this will be needed. Also necessary for measurement is metric scale, and the integration of an inertial sensor into this final SLAM system would provide both scale and some potential redundancy against visual navigation failures.

9.2.3 Scene Statistics and Learning Adaptations

The work on visual scene statistics for dataset characterisation tools presented in this thesis has already been extended by MSc project students to include colour spaces, spacial correlation and noise analysis, as well as evaluating two new challenging environments in underwater and surgical scenes. Adding more statistics would further expand the capability of these tests to determine environmental traits. These could be measuring statistics from visual ecology that haven't been included yet, such as range data, which impacts upon depth and motion coding (Geisler, 2008) and is plausibly different between human environments where structures such as walls dominate the edges of the scene and natural environments where they are spread throughout it. We might also integrate higher level understanding into the statistics, such as using semantic classification to look at the types of materials or objects present within a scene and how these relate to SLAM performance. This work would also benefit from alternative measures of existing statistics, especially for traits like in-scene motion or texture where there are a lot of potential confusing factors. Optical flow could form the basis for an alternative approach to measuring in-scen motion, for example.

The next step for this line of work would be to move from simply detecting the traits of environments to using them to make recommendations about system construction and calibration, as an extension of the design space exploration work done by Saeedi et al. (2017). By correlating visual scene statistics from a large number of datasets with performance of various SLAM systems, a tool could be developed whereby users can input example data from their intended application domain and be recommended a whole system, components such as feature descriptors, or configurations of parameters that are most likely to perform well. Alternatively with enough computation resources, measurement of scene statistics as a system runs could allow some real-time adaptation. A simple version of this might involve measuring scene statistics during the initialisation procedure for ORBSLAM and, in the presence of high texture, dynamically increasing the number of features extracted.

9.2.4 Simulation and Evaluation

Finally, the question of quality evaluation data for further forest navigation development remains an open one. There are still relatively few datasets for forests compared to traditional domains, despite the datasets produced by this thesis and the recent Finn-Forest dataset (Ali et al., 2020), but it looks like this will be an important goal of the Digiforest project (Digiforest-Team, 2022), which started in 2022. The field would benefit greatly from substantial work being invested in a forest dataset with both track and map ground truth, to enable evaluation of even the scene reconstruction capabilities of tested systems. This would doubtless require dense pre or post mapping of the environment with a lidar scanner, a vicon style MAV tracking setup sufficiently camouflaged to avoid spoiling the data, and significant post processing to ensure alignment. Evaluation on this dataset would still need to be aware that any amount of wind will have resulted in disagreement between camera and laser data.

While we have found that simulation is an imperfect alternative to real world data, tests on traits such as reconstruction quality and resilience to in-scene motion can still be performed much more accurately with a simulator. The visual scene statistics presented here have acted as a useful guide in continued simulation development, allowing quick feedback on whether improvements were reflected in increased similarity to real forest statistics. Future work could continue from the photorealistic simulator presented in Chapter 6 and attempt to improve the realism of the environment further, or find ways to have further fine tuning control over individual challenging traits in order to best test new SLAM developments.

Appendix A

Modular Vision System

Below is a description of all modules created for the EnVisionEd, the modular vision system used for running many of the experiments in this thesis.

Name	**Description**
BallFinder	Test module implementing code to find a red ball
BasicStats	Reporter module outputting minimum, maximum, mean and standard deviation of an input numeric value
BGRtoGrey	Outputs greyscale images from BGR inputs
BkgSubtractor	Outputs a version of an input image with its background subtracted by OpenCV's MOG2 background subtractor
BoxPlotter	A display / reporter for displaying box plots
CameraDisplayer	Displays an input image
CLAHE	Outputs a contrast limited adaptive histogram normalisation of an input image
ColourWheelOF	Displays input optical flow data on an input image using a conversion to colour
Contrast	Outputs the contrast value of an input image

Name	**Description**
Copy	Outputs a copy of the input data
Counter	Outputs the length of an input
Cropper	Outputs a cropped version of an input image
DenseOF	Outputs a dense optical flow field for an input greyscale image
Difference	Outputs the difference between an input numeric value and its previous value
FeatureDescriber	Outputs feature descriptions for the input keypoints of the input greyscale image. Feature choices for this are BRIEF, ORB, SIFT, SURF, BRISK
FeatureDetector	Outputs keypoints for the input greyscale image. Types available are Fast, Harris Corners, Dense or Censure
FeatureExtractor	Outputs keypoints and features for the input greyscale image. Features available are ORB, SIFT, SURF and BRISK
FeatureMatcher	Outputs feature matches for the input descriptors. Matching options are Brute Force or Brute Force with Ratio Test. This module can also display matches, if provided with the keypoints and greyscale image.
FileLoader	Reads Data from files
FileReporter	Outputs fields from Data to files
Framerate	Uses ”duration” inputs to calculate the current rate of the system
FrequencyDisplayer	Specialist graph dispayer for frequencies of occurence in an input list

Name	**Description**
GraphDisplayer	Displays graphs
Grapher	Helper class combining GraphDisplayer and Lister
GreyScaler	Converts an input image to greyscale
HistogramReporter	Displays histograms
HSVer	Converts an input image to HSV colour space
ImageBufferingDisplayer	Buffers input images an outputs the buffered image if no new input image is received
ImageOutput	Saves input image frames to the file system
ImageRecogniser	Outputs which previously seen image has the smallest MSE to the current input image
ImageReprojector	Reprojects an input image according to input keypoint matches
ImageSavingDisplay	Displays an input image and saves it to file if the space bar is pressed
ImageSelectiveDisplay	Displays an input image only if a criteria is met (eg. a classifier output)
ImageTextDisplay	Displays an input image with text overlaid at the bottom
IntensityHist	Outputs the intensity histogram of an input image
KLDivergence	Outputs the KL Divergence between two input images
Laplacian	Outputs the Laplacian of an input image
Lister	Appends an input value to a list in Data
LowPassFilterer	Outputs the filtered (wiener, gaussian or bilateral) version of an input image
Luminance	Outputs the mean luminance value of an input image
MeanSquaredError	Outputs the MSE between two input images

Name	**Description**
Multiplier	Outputs the defined multiple of an input value
Normaliser	Outputs a normalised version of an input greyscale image
PatchProcess	Splits an input image into smaller patches
Percentage	Converts an input numeric value to a percentage, given a defined maximum value
PlaceClassifier	Outputs the VGG16 classification of an input image
Resizer	Outputs a resized version of an input image
ROSPublisher	Publishes an image to a ROS topic
Rotator	Outputs a rotated version of an input image
StringCounter	Counts how many times a string appears in an input list
ThresholdClassifier	Outputs a classification of a numeric input based on a set of thresholds and classes
TimeKeeper	Outputs a timestamp and the duration since the last timestamp
Whitener	Performs ZCA whitening on greyscale images
WindowAverage	Outputs an average over a defined window size for any numeric value in Data
CameraFeed	Sensor that serves frames from a live camera feed or folder of frame files
ROSFeed	Sensor that serves frames recieved from a specified topic in a ROS bag file
ZipFeed	Extension of the camera feed sensor to handle files saved in zip archives at any level of nesting

Bibliography

Aguiar, A. S., dos Santos, F. N., Cunha, J. B., Sobreira, H., and Sousa, A. J. (2020). Localization and mapping for robots in agriculture and forestry: A survey. *Robotics*, 9(4):97.

Aladem, M., Cofield, A., and Rawashdeh, S. A. (2018). Image preprocessing for stereoscopic visual odometry at night. In *2018 IEEE International Conference on Electro/Information Technology (EIT)*, pages 0785–0789. IEEE.

Ali, I., Durmush, A., Suominen, O., Yli-Hietanen, J., Peltonen, S., Collin, J., and Gotchev, A. (2020). Finnforest dataset: A forest landscape for visual slam. *Robotics and Autonomous Systems*, 132:103610.

Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., and Sivic, J. (2016). Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5297–5307.

Asmar, D. C., Zelek, J. S., and Abdallah, S. M. (2004). SmartSLAM : localization and mapping across multi-environments. *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, pages 5240 – 5245.

Asmar, D. C., Zelek, J. S., and Abdallah, S. M. (2005). Seeing the trees before the forest [natural object detection]. In *The 2nd Canadian Conference on Computer and Robot Vision (CRV'05)*, pages 587–593. IEEE.

Asmar, D. C., Zelek, J. S., and Abdallah, S. M. (2006). Tree trunks as landmarks for outdoor vision slam. In *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*, pages 196–196. IEEE.

AutonomyLab (2014). Ardrone autonomy. `https://github.com/AutonomyLab/ardrone_autonomy`.

AutonomyLab (2018). Bebop autonomy. `https://github.com/AutonomyLab/bebop_autonomy`.

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer.

Bodin, B., Wagstaff, H., Saeedi, S., Nardi, L., Vespa, E., Mayer, J., Nisbet, A., Lujn, M., Furber, S., Davison, A., Kelly, P., and O'Boyle, M. (2018). Slambench2: Multi-objective head-to-head benchmarking for visual slam. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*.

Bruce, J., Wawerla, J., and Vaughan, R. (2015). The SFU mountain dataset: Semi-structured woodland trails under changing environmental conditions. In *IEEE Int. Conf. on Robotics and Automation 2015, Workshop on Visual Place Recognition in Changing Environments*.

Campos, C., Elvira, R., Rodríguez, J. J. G., Montiel, J. M., and Tardós, J. D. (2021). Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890.

Chahine, G. and Pradalier, C. (2018). Survey of monocular slam algorithms in natural environments. In *2018 15th Conference on Computer and Robot Vision (CRV)*, pages 345–352. IEEE.

Chen, X., Zhang, B., Luo, L., et al. (2018). Multi-feature fusion tree trunk detection and orchard mobile robot localization using camera/ultrasonic sensors. *Computers and Electronics in Agriculture*, 147:91–108.

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

Collier, J. and Ramirez-Serrano, A. (2009). Environment classification for indoor/outdoor robotic mapping. *Proceedings of the 2009 Canadian Conference on Computer and Robot Vision, CRV 2009*, pages 276–283.

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

Cummins, M. and Newman, P. (2011). Appearance-only slam at large scale with fab-map 2.0. *The International Journal of Robotics Research*, 30(9):1100–1123.

Davison, A. J. (2018). Futuremapping: The computational structure of spatial ai systems. *arXiv preprint arXiv:1803.11288*.

Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. (2007). Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067.

Digiforest-Team (2022).

Doersch, C. and Zisserman, A. (2019). Sim2real transfer learning for 3d human pose estimation: motion to the rescue. *Advances in Neural Information Processing Systems*, 32.

Duggal, V., Sukhwani, M., Bipin, K., Reddy, G. S., and Krishna, K. M. (2016). Plantation monitoring and yield estimation using autonomous quadcopter for precision agriculture. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 5121–5127. IEEE.

Engel, J., Koltun, V., and Cremers, D. (2017). Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*.

Engel, J., Schöps, T., and Cremers, D. (2014). LSD-SLAM: Large-Scale Direct Monocular SLAM. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision ECCV 2014*, volume 8690 of *Lecture Notes in Computer Science*, pages 834–849. Springer International Publishing, Cham.

Engel, J., Usenko, V., and Cremers, D. (2016a). A photometrically calibrated benchmark for monocular visual odometry. *arXiv preprint arXiv:1607.02555*.

Engel, J., Usenko, V., and Cremers, D. (2016b). A photometrically calibrated benchmark for monocular visual odometry. In *arXiv:1607.02555*.

Epic Games (2018). Unreal engine. `https://www.unrealengine.com`.

Fan, Y., Feng, Z., Shen, C., Khan, T. U., Mannan, A., Gao, X., Chen, P., and Saeed, S. (2020). A trunk-based slam backend for smartphones with online slam in large-scale forest inventories. *ISPRS Journal of Photogrammetry and Remote Sensing*, 162:41–49.

Forsman, M., Holmgren, J., and Olofsson, K. (2016). Tree stem diameter estimation from mobile laser scanning using line-wise intensity-based clustering. *Forests*, 7(9):206.

Forster, C., Pizzoli, M., and Scaramuzza, D. (2014). Svo: Fast semi-direct monocular visual odometry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 15–22. IEEE.

Gaidon, A., Wang, Q., Cabon, Y., and Vig, E. (2016). Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4340–4349.

Gálvez-López, D. and Tardós, J. D. (2012). Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197.

Garforth, J. and Webb, B. (2019). Visual appearance analysis of forest scenes for monocular slam. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1794–1800. IEEE.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Geisler, W. S. (2008). Visual Perception and the Statistical Properties of Natural Scenes.

Giusti, A., Guzzi, J., Cireşan, D. C., He, F.-L., Rodríguez, J. P., Fontana, F., Faessler, M., Forster, C., Schmidhuber, J., Di Caro, G., et al. (2015). A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2):661–667.

GmbH, M. U. (Last updated 2022). Broadleaf forest collection.

Gomez-Ojeda, R., Lopez-Antequera, M., Petkov, N., and Gonzalez-Jimenez, J. (2015). Training a convolutional neural network for appearance-invariant place recognition. *arXiv preprint arXiv:1505.07428*.

Gomez-Ojeda, R., Zhang, Z., Gonzalez-Jimenez, J., and Scaramuzza, D. (2018). Learning-based image enhancement for visual odometry in challenging hdr environments. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 805–811. IEEE.

Gómez-Rodríguez, J. J., Lamarca, J., Morlana, J., Tardós, J. D., and Montiel, J. M. (2021). Sd-defslam: Semi-direct monocular slam for deformable and intracorporeal scenes. In *2021 IEEE international conference on robotics and automation (ICRA)*, pages 5170–5177. IEEE.

Handa, A., Whelan, T., McDonald, J., and Davison, A. J. (2014). A benchmark for rgb-d visual odometry, 3d reconstruction and slam. In *2014 IEEE international conference on Robotics and automation (ICRA)*, pages 1524–1531. IEEE.

Harris, C. G., Stephens, M., et al. (1988). A combined corner and edge detector. In *Alvey vision conference*, pages 10–5244. Citeseer.

Hausler, S., Garg, S., Xu, M., Milford, M., and Fischer, T. (2021). Patch-netvlad: Multi-scale fusion of locally-global descriptors for place recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14141–14152.

He, M., Zhu, C., Huang, Q., Ren, B., and Liu, J. (2020). A review of monocular visual odometry. *The Visual Computer*, 36(5):1053–1065.

Holmes, S. A. and Murray, D. W. (2012). Monocular slam with conditionally independent split mapping. *IEEE transactions on pattern analysis and machine intelligence*, 35(6):1451–1463.

Hfer, S., Bekris, K., Handa, A., Gamboa, J. C., Mozifian, M., Golemo, F., Atkeson, C., Fox, D., Goldberg, K., Leonard, J., Karen Liu, C., Peters, J., Song, S., Welinder, P., and White, M. (2021). Sim2real in robotics and automation: Applications and challenges. *IEEE Transactions on Automation Science and Engineering*, 18(2):398–400.

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

Johns, E. and Yang, G.-Z. (2013). Feature co-occurrence maps: Appearance-based localisation throughout the day. In *2013 IEEE International Conference on Robotics and Automation*, pages 3212–3218. IEEE.

Jonas, J. B., Schneider, U., and Naumann, G. O. (1992). Count and density of human retinal photoreceptors. *Graefe's Archive for Clinical and Experimental Ophthalmology*, 230(6):505–510.

Kadian, A., Truong, J., Gokaslan, A., Clegg, A., Wijmans, E., Lee, S., Savva, M., Chernova, S., and Batra, D. (2020). Sim2real predictivity: Does evaluation in simulation predict real-world performance? *IEEE Robotics and Automation Letters*, 5(4):6670–6677.

Kendall, A., Grimes, M., and Cipolla, R. (2015). Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946.

Klein, G. and Murray, D. (2007). Parallel Tracking and Mapping for Small AR Workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10. IEEE, IEEE.

Kukko, A., Kaijaluoto, R., Kaartinen, H., Lehtola, V. V., Jaakkola, A., and Hyyppä, J. (2017). Graph slam correction for single scanner mls forest data under boreal forest canopy. *ISPRS journal of photogrammetry and remote sensing*, 132:199–209.

Lamarca, J. and Montiel, J. M. M. (2018). Camera tracking for slam in deformable maps. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0.

Laughlin, S. (1981). A simple coding procedure enhances a neuron's information capacity. *Zeitschrift für Naturforschung c*, 36(9-10):910–912.

Layne, J., Land, M., and Zeil, J. (1997). Fiddler crabs use the visual horizon to distinguish predators from conspecifics: a review of the evidence. *Journal of the Marine Biological Association of the United Kingdom*, 77(1):43–54.

Lefsky, M. A., Cohen, W. B., Acker, S. A., Parker, G. G., Spies, T. A., and Harding, D. (1999). Lidar remote sensing of the canopy structure and biophysical properties of Douglas-fir western hemlock forests. *Remote Sensing of Environment*, 70(3):339–361.

Leutenegger, S., Chli, M., and Siegwart, R. Y. (2011). Brisk: Binary robust invariant scalable keypoints. In *2011 International conference on computer vision*, pages 2548–2555. Ieee.

Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., and Furgale, P. (2015). Keyframe-based visual-inertial odometry using nonlinear optimization. *International Journal of Robotics Research*, 34(3):314–334.

Li, X., Wang, S., Zhao, Y., Verbeek, J., and Kannala, J. (2019). Hierarchical scene coordinate classification and regression for visual localization.

Liao, F., Lai, S., Hu, Y., Cui, J., Wang, J. L., Teo, R., and Lin, F. (2016). 3d motion planning for uavs in gps-denied unknown forest environment. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 246–251. IEEE.

Liu, X., Nardari, G. V., Ojeda, F. C., Tao, Y., Zhou, A., Donnelly, T., Qu, C., Chen, S. W., Romero, R. A., Taylor, C. J., et al. (2022). Large-scale autonomous flight with real-time semantic slam under dense forest canopy. *IEEE Robotics and Automation Letters*, 7(2):5512–5519.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

Lowry, S. and Andreasson, H. (2018). Lightweight, viewpoint-invariant visual place recognition in changing environments. *IEEE Robotics and Automation Letters*, 3(2):957–964.

Lynen, S., Bosse, M., Furgale, P., and Siegwart, R. (2014). Placeless place-recognition. In *2014 2nd International Conference on 3D Vision*, volume 1, pages 303–310. IEEE.

Lynen, S., Bosse, M., and Siegwart, R. (2017). Trajectory-based place-recognition for efficient large scale localization. *International Journal of Computer Vision*, 124:49–64.

Maciel-Pearson, B. G., Carbonneau, P., and Breckon, T. P. (2018). Extending deep neural network trail navigation for unmanned aerial vehicle operation within the forest canopy. In *TAROS*.

Maciel-Pearson, B. G., Marchegiani, L., Akay, S., Atapour-Abarghouei, A., Garforth, J., and Breckon, T. P. (2019). Online deep reinforcement learning for autonomous uav navigation and exploration of outdoor environments. *ArXiv*, abs/1912.05684.

Maini, R. and Aggarwal, H. (2009). Study and comparison of various image edge detection techniques. *International journal of image processing (IJIP)*, 3(1):1–11.

McGlade, J., Wallace, L., Reinke, K., and Jones, S. (2022). The potential of low-cost 3d imaging technologies for forestry applications: Setting a research agenda for low-cost remote sensing inventory tasks. *Forests*, 13(2):204.

Miettinen, M., Ohman, M., Visala, A., and Forsman, P. (2007). Simultaneous localization and mapping for forest harvesters. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 517–522. IEEE.

Milford, M. (2013). Vision-based place recognition: how low can you go? *The International Journal of Robotics Research*, 32(7):766–789.

Milford, M. and Wyeth, G. F. (2012). SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1643–1649.

Moore, R. J., Thurrowgood, S., Soccol, D., Bland, D., and Srinivasan, M. V. (2010). A bio-inspired stereo vision system for guidance of autonomous aircraft. *Advances in Theory and Applications of Stereo Vision*.

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

Mur-Artal, R. and Tardós, J. D. (2015). Probabilistic semi-dense mapping from highly accurate feature-based monocular slam. In *Robotics: Science and Systems*, volume 2015. Rome.

Mur-Artal, R. and Tardós, J. D. (2017). Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262.

Newcombe, R., Davison, A. J., Izadi, S., Kohli, P., Hilliges, O., Shotton, J., Molyneaux, D., Hodges, S., Kim, D., and Fitzgibbon, A. (2011). KinectFusion: Real-time dense surface mapping and tracking. *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136.

Niu, C., Newlands, C., Zauner, K.-P., and Tarapore, D. (2023). An embarrassingly simple approach for visual navigation of forest environments. *Frontiers in Robotics and AI*, 10.

of Edinburgh Council, C. (2017). Corstorphine hill local nature reserve management plan 2017 2026. `https://consultationhub.edinburgh.gov.uk/sfc/corstorphine-hill-lnr/supporting_documents/Corstorphine%20Hill%20Mgt%20Plan%202017%20%202026.pdf`.

Ohman, M. and Kosti Kannas, Jaakko Jutila, Arto Visala and Pekka Forsman, M. M. (2008). Tree Measurement and Simultaneous Localization and Mapping System for Forest Harvester. *Field and Service Robotics Springer Tracts in Advanced Robotics*, 42:369–378.

Olid, D., Fácil, J. M., and Civera, J. (2018). Single-view place recognition under seasonal changes. *arXiv preprint arXiv:1808.06516*.

Park, S., Pahk, J., Jahn, L. L. F., Lim, Y., An, J., and Choi, G. (2023). A study on quantifying sim2real image gap in autonomous driving simulations using lane segmentation attention map similarity. In *International Conference on Intelligent Autonomous Systems*, pages 203–212. Springer.

Pierzchała, M., Giguère, P., and Astrup, R. (2018). Mapping forests using an unmanned ground vehicle with 3d lidar and graph-slam. *Computers and Electronics in Agriculture*, 145:217–225.

Qin, T., Li, P., and Shen, S. (2018). Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan.

Risse, B., Mangan, M., Stürzl, W., and Webb, B. (2018). Software to convert terrestrial lidar scans of natural environments into photorealistic meshes. *Environmental Modelling & Software*, 99:88–100.

Rosen, D. M., Doherty, K. J., Terán Espinoza, A., and Leonard, J. J. (2021). Advances in inference and representation for simultaneous localization and mapping. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:215–242.

Ross, S., Melik-Barkhudarov, N., Shankar, K. S., Wendel, A., Dey, D., Bagnell, J. A., and Hebert, M. (2013). Learning monocular reactive uav control in cluttered natural environments. In *2013 IEEE international conference on robotics and automation*, pages 1765–1772. IEEE.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. *Proceedings of the IEEE International Conference on Computer Vision*, pages 2564–2571.

Sabo, C. M., Cope, A., Gurney, K., Vasilaki, E., and Marshall, J. (2016). Bio-inspired visual navigation for a quadcopter using optic flow. In *AIAA Infotech@ Aerospace*, page 0404.

Sadeghi, F., Toshev, A., Jang, E., and Levine, S. (2018). Sim2real viewpoint invariant visual servoing by recurrent control. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Saeedi, S., Nardi, L., Johns, E., Bodin, B., Kelly, P., and Davison, A. (2017). Application-oriented design space exploration for slam algorithms. pages 5716–5723.

Schultz, A., Gilabert, R., Bharadwaj, A., de Haag, M. U., and Zhu, Z. (2016). A navigation and mapping method for uas during under-the-canopy forest operations. In *2016 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 739–746. IEEE.

Scona, R., Jaimez, M., Petillot, Y. R., Fallon, M., and Cremers, D. (2018). Staticfusion: Background reconstruction for dense rgb-d slam in dynamic environments. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626.

Shah, S., Dey, D., Lovett, C., and Kapoor, A. (2017). Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*.

Shu, F., Lesur, P., Xie, Y., Pagani, A., and Stricker, D. (2020).

Silva, A., Mendonça, R., and Santana, P. (2020). Monocular trail detection and tracking aided by visual slam for small unmanned aerial vehicles. *Journal of Intelligent & Robotic Systems*, 97(3):531–551.

Simoncelli, E. P. and Olshausen, B. A. (2001). Natural Image Statistics and Neural Representation. *Annual Review Neuroscience*.

Smolyanskiy, N., Kamenev, A., Smith, J., and Birchfield, S. (2017). Toward low-flying autonomous mav trail navigation using deep neural networks for environmental awareness. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4241–4247. IEEE.

Somanathan, H., Warrant, E. J., Borges, R. M., Wallén, R., and Kelber, A. (2009). Resolution and sensitivity of the eyes of the asian honeybees apis florea, apis cerana and apis dorsata. *Journal of Experimental Biology*, 212(15):2448–2453.

Srinivasan, M. V., Laughlin, S. B., and Dubs, A. (1982). Predictive coding: a fresh view of inhibition in the retina. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 216(1205):427–459.

Stone, T., Differt, D., Milford, M., and Webb, B. (2016). Skyline-based localisation for aggressively manoeuvring robots using uv sensors and spherical harmonics. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5615–5622. IEEE.

Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580. IEEE.

Sun, P. and Lau, H. (2019). Rgb-channel-based illumination robust slam method. *Journal of Automation and Control Engineering*, 7(2).

Sünderhauf, N., Neubert, P., and Protzel, P. (2013). Are we there yet? challenging seqslam on a 3000 km journey across all four seasons. In *Proc. of Workshop on Long-Term Autonomy, IEEE International Conference on Robotics and Automation (ICRA)*, page 2013.

Szot, A., Clegg, A., Undersander, E., Wijmans, E., Zhao, Y., Turner, J., Maestre, N., Mukadam, M., Chaplot, D., Maksymets, O., Gokaslan, A., Vondrus, V., Dharur, S., Meier, F., Galuba, W., Chang, A., Kira, Z., Koltun, V., Malik, J., Savva, M., and Batra, D. (2021). Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Ta, D.-N., Ok, K., and Dellaert, F. (2013). Monocular Parallel Tracking and Mapping with Odometry Fusion for MAV Navigation in Feature-lacking Environments. *Intelligent Robots and Systems (. . . .*

Takashi, T., Asuka, A., Toshihiko, M., Shuhei, K., Keiko, S., Mitsuhiro, M., Shuhei, T., Shuichi, N., Akiko, M., Yukihiro, C., Kouji, S., and Toru, H. (2014). Forest 3D Mapping and Tree Sizes Measurement for Forest Management Based on Sensing Technology for Mobile Robots. *Springer Tracts in Advanced Robotics*, 92:357–368.

Tang, J., Chen, Y., Kukko, A., Kaartinen, H., Jaakkola, A., Khoramshahi, E., Hakala, T., Hyyppä, J., Holopainen, M., and Hyyppä, H. (2015). Slam-aided stem mapping for forest inventory with small-footprint mobile lidar. *Forests*, 6(12):4588–4606.

Teed, Z. and Deng, J. (2021). Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Advances in neural information processing systems*, 34:16558–16569.

Tian, Y., Liu, K., Ok, K., Tran, L., Allen, D., Roy, N., and How, J. P. (2020). Search and rescue under the forest canopy using multiple uavs. *The International Journal of Robotics Research*, 39(10-11):1201–1221.

Tinchev, G., Nobili, S., and Fallon, M. (2018). Seeing the wood for the trees: Reliable localization in urban and natural environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8239–8246. IEEE.

Tremblay, J.-F., Béland, M., Gagnon, R., Pomerleau, F., and Giguère, P. (2020). Automatic three-dimensional mapping for tree diameter measurements in inventory operations. *Journal of Field Robotics*.

Vonikakis, V., Chrysostomou, D., Kouskouridas, R., and Gasteratos, A. (2012). Improving the robustness in feature detection by local contrast enhancement. In *2012 IEEE International Conference on Imaging Systems and Techniques Proceedings*, pages 158–163. IEEE.

Vysotska, O. and Stachniss, C. (2019). Effective visual place recognition using multi-sequence maps. *IEEE Robotics and Automation Letters*, 4(2):1730–1736.

Wakakuwa, M., Kurasawa, M., Giurfa, M., and Arikawa, K. (2005). Spectral heterogeneity of honeybee ommatidia. *Naturwissenschaften*, 92(10):464–467.

Walch, F., Hazirbas, C., Leal-Taixe, L., Sattler, T., Hilsenbeck, S., and Cremers, D. (2017). Image-based localization using lstms for structured feature correlation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 627–637.

Wang, W., Zhu, D., Wang, X., Hu, Y., Qiu, Y., Wang, C., Hu, Y., Kapoor, A., and Scherer, S. (2020). Tartanair: A dataset to push the limits of visual slam. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4909–4916.

Wikipedia contributors (2023). Corstorphine hill — Wikipedia, the free encyclopedia. [Online; accessed 22-July-2004].

Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., and Tardós, J. (2009). A comparison of loop closing techniques in monocular slam. *Robotics and Autonomous Systems*, 57(12):1188–1197.

Williams, B., Klein, G., and Reid, I. (2011). Automatic relocalization and loop closing for real-time monocular slam. *IEEE transactions on pattern analysis and machine intelligence*, 33(9):1699–1712.

Yang, N., Wang, R., Gao, X., and Cremers, D. (2018). Challenges in monocular visual odometry: Photometric calibration, motion bias, and rolling shutter effect. *IEEE Robotics and Automation Letters*, 3(4):2878–2885.

Yi, K. M., Trulls, E., Lepetit, V., and Fua, P. (2016). Lift: Learned invariant feature transform. In *European Conference on Computer Vision*, pages 467–483. Springer.

Zhang, X., Shen, P., Luo, L., Zhang, L., and Song, J. (2012). Enhancement and noise reduction of very low light level images. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 2034–2037. Ieee.

Zhang, Z. and Scaramuzza, D. (2018). A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7244–7251. IEEE.

Zollikofer, C., Wehner, R., and Fukushi, T. (1995). Optical scaling in conspecific cataglyphis ants. *Journal of Experimental Biology*, 198(8):1637–1646.