Hyperparameter Optimization for SLAM: An Approach For Enhancing ORB-SLAM2's Performance

by

Eduardo Ismael Montemayor Castillo

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science
University of Alberta

Abstract

Simultaneous Location and Mapping (SLAM) has been a well-pursued research area for computer vision and robotics. Robustness and performance are fields that address the efficiency of SLAM solutions. Hyperparameter Optimization (HPO) promises to find a hyperparameter set that displays the lowest error within a validation set. This thesis aims to devise a methodology that applies HPO to SLAM to reduce the absolute trajectory error produced and to increase performance by building a more accurate map. Specifically, it investigates whether the proposed methodology impacts error reduction on ORB-SLAM2. We train model-free, population-based algorithms in a modified KITTI benchmark to obtain an initial set of possible configurations and test them against model-free, search-based baseline algorithms. We used a combination of 20 modified and unaltered sequences for performance evaluation. Four evaluation metrics (optimality, proximity, under-performance, and success rates) determine the efficacy of each candidate configuration. The proposed methodology outperformed a default configuration execution with an 80 % success rate. The results promise case-specific executions. However, we could not find a universal hyperparameter set to reduce error in all test cases. The proposed methodology has a simple implementation, is cost-effective, does not need an expert tuner, and shows up to 60 % error reduction.

Preface

This thesis and its methodology (Chapter 3) collaborate with Dr. Hong Zhang. Shortened sequence training and testing described in Chapter 3 and Chapter 4 are designed by the author with the assistance of Dr. Zhang. The background information and literature review commentary (Chapter 2), experiments and results (Chapter 3), and conclusions (Chapter 5) are original works.

"There are no secrets to success. It is the result of preparation, hard work, and learning from failure"

- Colin Powell, former U.S. Defense Secretary

To my parents and grandparents:

For believing in my hard work and life-changing decisions.

Thank you for your never-ending support.

Acknowledgements

First and foremost, I want to thank my supervisor Dr. Hong Zhang for his guidance throughout my studies. He has been very patient, understanding, and encouraging. His mentorship has taught me to research, think out of the box, and question the literature.

I want to thank the Robotics and Vision lab members, especially Islam Ali, Moein Shakeri, Shing-Yan, Ehsan Ahmadi, and Ali Jahani, for their help, support, and friendship throughout this journey. I would reach out for guidance and opinions whenever I hit a roadblock. There was always someone who would question my research and push me to polish my work.

I want to thank my friend Junaid Ahmad for sticking around despite the 2020 pandemic. Another important person is my partner, who has shown endless support throughout my studies. Without her, I would not have known about the country's beauty. She also helped improve my writing and presentation skills. I would not have believed that I would find her amidst a pandemic by a coincidental idea.

Another important person I am grateful to meet is Dr. Ioanis Nikolaidis. He was patient during the coursework studies and taught that literature is not flawless. It has assumptions and hidden messages leading to limitations within the presented information.

Finally, I would like to thank my parents and grandparents back in Mexico, who have provided endless support. They have been worried sick during the 2020 crisis and have never stopped looking out for me. I would not have arrived at my current position without their teachings. I am very fortunate to have the privilege to study

abroad and expand my future. They believed in me, and I will not let them down. My only wish is to give society the knowledge and gifts I have acquired.

Table of Contents

List of Tables

List of Figures

Abbreviations

ATE - Absolute Trajectory Error.

BO - Bayesian Optimization.

BOHB - Bayesian Optimization and HyperBand.

BRIEF - Binary Robust Independent Elementary Features.

CI - Confidence Interval.

EA - Evolutionary Algorithm.

FAST - Features from Accelerated Segment Test.

GA - Genetic Algorithm.

HB - HyperBand.

HPO - Hyperparameter Optimization.

HPT - Hyperparameter Tuning.

ML - Machine Learning.

ORB - Oriented FAST and rotated BRIEF.

PSO - Particle Swarm Optimization.

RMSE - Root-Mean Square Error.

RNG - Random Number Generator.

SA - Simulated Annealing.

SH - Successive Halving.

SLAM - Simultaneous Location and Mapping.

SVM - Support Vector Machine.

VO - Visual Odometry.

VSLAM - Visual Simultaneous Location and Mapping.

Chapter 1

Introduction

Simultaneous Location and Mapping (SLAM) and Visual Odometry (VO) are the fundamental constituents of emerging modern-day technologies for robotics research. Many applications use SLAM and VO, such as the navigation of crewless vehicles and the boom of virtual and augmented reality. In recent years, the study of SLAM and visual SLAM (VSLAM) solutions have progressed towards real-time applications. These use either feature-based (indirect), direct, sparse, or dense approaches. The SLAM problem [1–4] has many potential solutions. Two state-of-the-art approaches are Oriented FAST (Features from Accelerated Segment Test) and Rotated BRIEF (Binary Robust Independent Elementary Features) feature detector, ORB-SLAM, [2] and its more robust version, ORB-SLAM2 [3].

Autonomous navigation is a well-pursued research area for mobile robotics. Many factors, environments, and situations affect a robot's performance, including: disaster scenes, maritime exploration, air surveillance, rescue, or GPS-denied locations. Hence, there is a critical need for accurate maps and pose estimation. A significant amount of research on map densification and SLAM robustness [4–9] exists, but far fewer studies on the impact of a VSLAM algorithm's hyperparameters and their effect on mapping performance have been made.

ORB-SLAM, as an example, has a default set of commonly-used hyperparameters. These hyperparameters are seldom modified when mapping due to their high performance. Default parameters are preferred because parameter tuning is a time-consuming process. The computational costs of a parameter space exploration can be exponential and, therefore, counterproductive to the overall tuning results. Also, it can be a complex procedure. Nonetheless, there are studies done regarding parameter tuning to increase SLAM performance [10–12].

All existent approaches (at the time of this work) share the goal of solving the SLAM problem with different robustness, pre-processing, and accuracy levels. However, they all possess the same fundamental shortcoming: they are optimizable. We present a methodology based on hyperparameter optimization as an approach to increase SLAM performance through the use of model-free, population-based algorithms. It is independent of the SLAM solution chosen and focuses on the influence of SLAM parameters on the trajectory error produced.

1.1 Motivation

In recent years, studies have presented solutions that increase the robustness and efficiency of SLAM [13]. The research is divided into map densification [6, 7, 9, 14], pose estimation [15–18], sensor fusion [7, 19], and visual odometry [3, 13, 20]. SLAM performance is dependent on trajectory estimation. Most open-source SLAM solutions available run with default parameter values. These are generally effective (i.e., produce an adequate trajectory estimation) but are not optimal for case-specific situations.

Research done in SLAM parameter tuning shows an increase in a given algorithm's performance if proper parameter values are applied [10, 12, 21–23]. Furthermore, optimization techniques are adaptable and compatible with SLAM [12, 23]. Applying these hyperparameter optimization algorithms in SLAM alludes to a significant performance increment.

1.2 Limitations

There are many different methods to execute SLAM. The most popular use either monocular, RGB-D, or stereo cameras. In this thesis, we will be focusing on applying model-free, population-based hyperparameter optimization algorithms in Stereo SLAM [24]. That is, the execution of SLAM using stereo cameras. Stereo cameras provide a better depth estimation than monocular ones, but do not have a complete depth map like the RGB-D option.

Although there are several other state-of-the-art SLAM solutions, we selected ORB-SLAM2 as the forerunner of SLAM optimization. Despite having slow initialization, tracking recovery issues, and point loss, it has a simpler standalone implementation compared to others. Due to the computational costs, we optimize a select number of parameters for a delimited[1] parameter space.

1.3 Contributions

The contributions of this thesis are as follows:

1. We propose a methodology that combines model-free, population-based hyperparameter optimization algorithms and SLAM. It is designed in such a way that it can be exported and used in any SLAM solution available[2].

2. We study the influence of a SLAM solution's parameters on the resulting performance. We apply a simple method for discerning which parameters have a higher correlation concerning the trajectory error.

3. We provide an implementation of different search-based and population-based optimization algorithms for obtaining SLAM parameter configurations.

[1]Having fixed boundaries or limits.
[2]Some adaptations may be needed.

1.4 Outline

The organization of the rest of this thesis is as follows.

Chapter 2 defines the concepts used throughout this thesis. It also presents a literature review on relevant research on Hyperparameter Optimization in SLAM.

Chapter 3 describes a methodology for optimizing visual SLAM solutions, consisting of an environmental setup, optimization constraints and assumptions, and a parameter selection criterion. It presents the selected population-based algorithms, the baseline search-based algorithms, and the training and testing benchmark.

Chapter 4 presents the experimental setup used. It proposes a three-step process for better-performing configuration[3] determination. The chapter shows the experimental results for each test suite.

Chapter 5 concludes this thesis and discusses future work.

[3] A comparison of the performance of these configurations and the default parameter execution determines if a hyperparameter set is more efficient than a default run.

Chapter 2

Background

This chapter aims to present the concepts of simultaneous location and mapping, absolute trajectory error, and hyperparameter optimization. Section 2.1 covers a general definition of SLAM, some existing algorithms in addition to ORB-SLAM2 (Section 2.1.1), and general applications. Section 2.2 focuses on a performance metric for evaluating SLAM solutions. Section 2.3 and subsequent subsections cover hyperparameter optimization, the five main branching categories, and popular algorithms. Section 2.9 presents a brief chapter summary of the concepts and research applicable to this thesis.

2.1 Simultaneous Location and Mapping

SLAM is vital for autonomous systems and navigation. It comprises the simultaneous estimation of a robot system with onboard sensors and constructing a model of the perceived environment [13, 15]. In simple terms, the robot's pose, which consists of direction, orientation, calibration parameters, and any other helpful sensor readings, describes its state. The model or map produced represents the environmental descriptors of the operation area. Maps produced by SLAM are essential. The maps serve as crucial information for the robot during path planning and function as a limiter for the error obtained during state estimation [13].

There are several different SLAM solutions, some of which do not involve a camera.

Visual SLAM (VSLAM) refers to a specific type of SLAM that leverages *3-D* vision to perform location and mapping functions when neither the environment nor the sensor's location are known [25]. This approach is crucial for autonomous navigation; it captures information with a camera (monocular [26], stereo [27], omnidirectional [28], time of flight [29], RGB-D[1] [30]) to determine important landmarks that aid in the mapping [31]. Then, the SLAM solutions use models to correlate images taken at different times to create *3-D* information about the landmarks' features and localize the robot [15].

SLAM solutions are mainly composed of the sensor-dependent front-end and the back-end. The former refers to a module that pre-processes data by extracting relevant features from the sensor data, by pixel extraction from images, and by data association from the landmarks observed [13]. The latter focuses on map estimation by treating the localization as a series of robot states, map optimization, keyframe estimation, maintenance, and global drift reduction by loop closure [6, 13]. The accuracy of a solution is dependent on the accuracy of its localization [32]. Figure 2.1 shows a representation of a typical SLAM system.

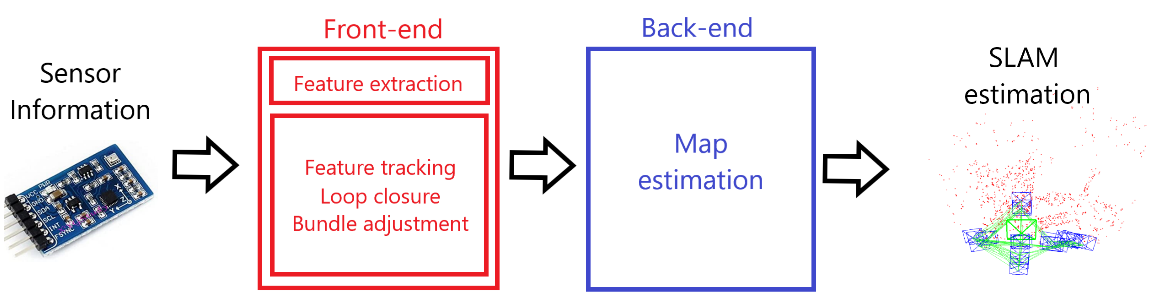


Figure 2.1: Representation of the front-end and back-end of a SLAM system. The back-end can provide feedback for loop closure to the front-end. Adapted from (Cadena, et al., 2016).

SLAM solutions can be divided depending on the type of approach used within the system. Regardless, the algorithms employ a probabilistic model that takes noisy measurements and computes an estimator (typically a maximum likelihood approach)

[1] Combined color and depth

for the unknown [5]. Solutions are usually a combination of two categories: *indirect* or *direct* and *sparse* or *dense*. The former refers to using sensor data in the probabilistic model. The latter uses the pixels in the images to construct a map.

Indirect methods are feature-based approaches that take raw sensor measurements and pre-process them to generate an intermediate representation. Then, those computed values are interpreted as noisy measurements in a probabilistic model to estimate the scene geometry[2] and camera motion [5]. Generally, indirect methods focus on minimizing the feature reprojection error [3]. ORB-SLAM [2–4] and RTAB-MAP [7] are example solutions implementing this method.

Direct methods jointly estimate motion and correspondences by minimizing the photometric error in direct image alignment [6]. This enables the use all the information from the input data [5]. These methods have higher accuracy and robustness, particularly in environments with little keypoints [14]. Additionally, this provides substantially more information about the geometry of the surroundings. Direct Sparse Odometry (DSO) [14] and Direct Sparse Odometry with Loop Closure (LDSO) [6] are example solutions implementing this method.

Sparse methods use and reconstruct only a selected number of points in *3-D*. They contain information about geometry, but not about the semantics of the scene [33]. Spare methods typically focus on the corners of images. This formulation has no notion of a neighborhood and keypoint positions are intrinsic and conditionally independent of camera poses [14]. ORB-SLAM [2, 3] (Figure 2.2) is an example solution that creates sparse maps.

Dense methods attempt to use and reconstruct all pixels contained within a *2-D* image. They exploit the connectedness of the used image region to formulate geometry priors and favor smoothness [14]. These priors allow passive vision to observe the dense world without computational aid. DSO [14], LDSO [6], and RTAB-MAP [7] (Figure 2.3) are example dense solutions.

[2]The representation of the *3-D* environment

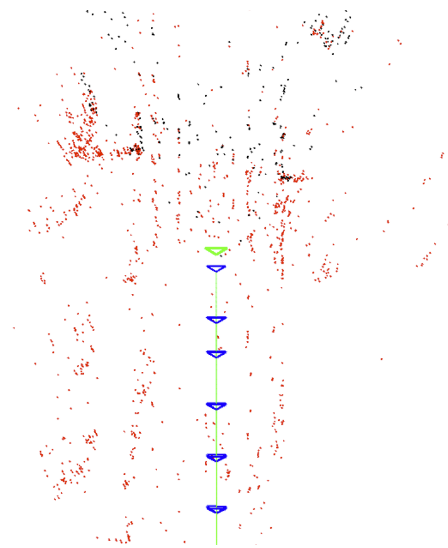


Figure 2.2: An example of a sparse map produced by ORB-SLAM2 by executing a KITTI sequence.

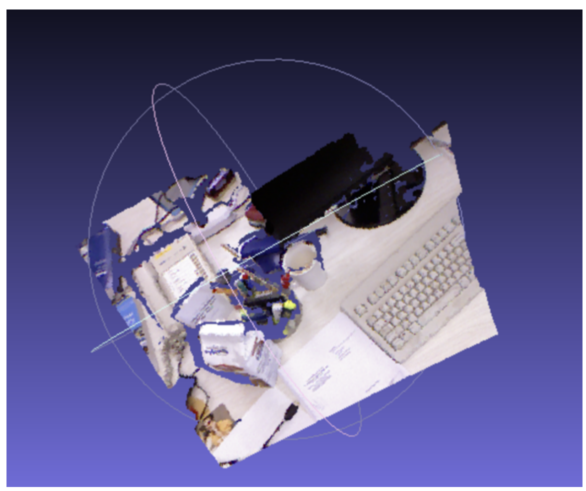


Figure 2.3: An example of a dense map and trajectory estimation produced by RTAB-MAP on the TUM Freiburg dataset.

Navigation has been a field of interest in SLAM for the last decade. SLAM relies either on feature matching [7] or visual odometry [2] to estimate robot poses and generate accurate maps. Estimating motion dynamics and triangulation from known landmarks are the basis for constructing these maps [34]. The map information is updated as new observations are acquired. Recent pose estimation studies have relied heavily on stereo cameras for depth perception [18], while others use different cameras to produce better maps (polarization [9, 19], monocular [2, 5, 35], or sensor fusion [36–39]). Indirect methods are computationally favorable to address navigation because optimization happens over the features and poses. These methods have reasonable pose estimations but produce sparse maps due to their dependence on selected features. Direct methods match the corresponding frames without rejecting any points and optimize the robot poses, which leads to dense maps with less accuracy.

There are many different graph-based solutions for SLAM. RTAB-Map[3] [7, 40] uses depth and RGB images to construct maps. Using the corresponding odometric poses embedded in the depth images and the transformations between each node of the graph created, RTAB-Map compares the images, performs a loop closure analysis, and optimizes the chart. The depth images generate a point cloud for each node and transform it using the poses. LSD-SLAM [5] is a direct feature-less monocular SLAM algorithm that allows building a large-scale, consistent map of the environment with highly accurate pose estimations based on direct image alignment. It reconstructs the landmarks in real-time as a pose-graph of keyframes with associated semi-dense depth maps obtained by filtering over many pixel-wise small-baseline stereo comparisons. DSO [14] and LDSO [6] are visual odometry-based systems that combine a straightforward probabilistic model with parameter optimization. These methods do not depend on keypoint detectors or descriptors, thereby allowing pixel sampling across all image regions that present intensity gradients. Thus, they achieve denser maps than those obtained from other SLAM solutions.

Although there are several other state-of-the-art SLAM solutions, we will focus exclusively on ORB-SLAM2 [2]. Despite having some flaws (e.g., slow initialization, tracking recovery issues, and point loss), its compatibility with the popular available datasets and its potential makes it our choice of a research subject. The following section will present a more thorough overview of the algorithm.

[3]Real Time Appearance Based Mapping.

2.1.1 ORB-SLAM2

ORB-SLAM2 is a SLAM solution that operates in real-time. It is compatible with monocular, stereo, and RGB-D cameras. It builds on its predecessor ORB-SLAM [2, 13, 15] by adding loop closing, map recycling, and relocalization capabilities [2]. The structure consists of three main threads that work in parallel: tracking, local mapping, and loop closure (which triggers a fourth thread that performs a full bundle adjustment after loop closure) [2, 15]. Figure 2.4 depicts the structure of ORB-SLAM2.

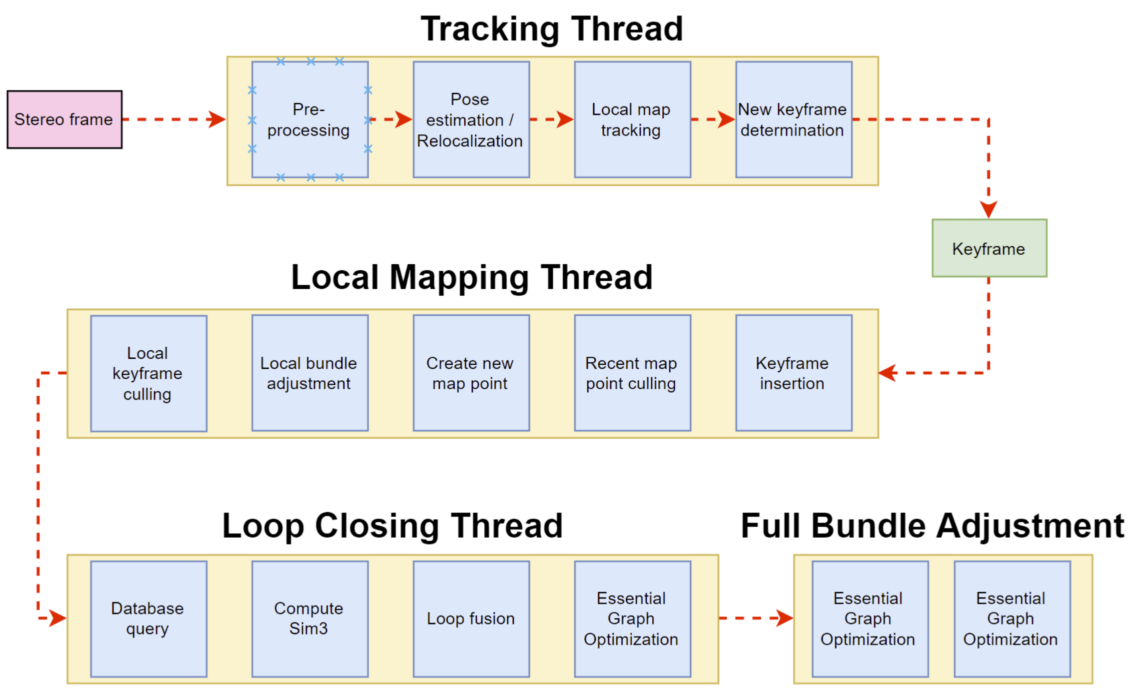


Figure 2.4: A visual representation of the ORB-SLAM2 algorithm. Adapted from (Andersson & Baerveldt, 2018)

For monocular cameras, the input extracts ORB features from a set of images. The FAST feature extraction finds edge features in the input frame. Then, the BRIEF, which represents the features as a string of binary numbers evaluated based on different pixel intensities between them, creates a descriptor [15, 41]. In the case of stereo cameras (used in this research), there is an ORB feature extraction from the left and right images. The system treats the left image as the reference for feature matching [15]. These ORB features, coupled with depth information, enable *3-D* mapping without any triangulation [3]. Then, the different threads use the information provided. Figure 2.5 is a representation of image pre-processing for ORB-SLAM2.

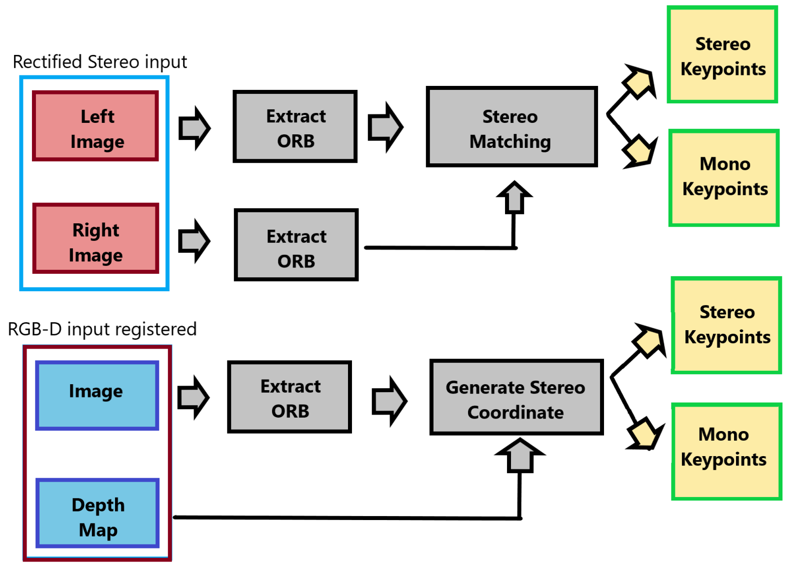


Figure 2.5: A visual representation of image input and pre-processing in ORB-SLAM2. Adapted from (Mur-Artal & Tardós, Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras, 2017)

The **tracking** thread consists of matching the local map with extracted features for every frame and minimizing the reprojection error to localize the camera by applying motion-only bundle adjustment [3]. It uses each frame for camera localization [15] and keyframe selection to construct a map. The thread finds the initial pose estimation by matching features between the current keyframe and the previous one. Then, it optimizes the current pose with a motion model to predict the corresponding map points on the keyframe and obtain an optimal *3-D* reconstruction [2, 15]. If tracking is lost, the system triggers the place recognition module to perform global relocalization. The module compares a new keyframe with the current frame to find a map fit with the largest inliers [15].

The **local mapping** thread manages and optimizes the local map by performing a local bundle adjustment [3]. It uses the acquired keyframes and map points to build a map. These keyframes are placed in a covisibility graph as nodes while treating common map points as edges [15]. Culling takes place if keyframes share many similarities with others and remove map points if too few keyframes observe them at a given time. This culling prevents unbounded growth of the covisibility graph and control keyframe redundancy.

The **loop closing** thread detects large loops and corrects the accumulated drift by performing pose graph optimization [2] using g2o[4] [42]. G2o achieves optimization by looking at the covisibility graph and the similarity between keyframes. It computes a score by examining the binary *bag-of-words* representation of the keyframe [15]. If a new keyframe shares more similarities with an existing keyframe than its neighbors, it is considered a loop candidate. The thread finds a loop if three of these candidates are connected. Then, the covisibility graph is updated accordingly [3, 15]. After the pose-graph optimization, ORB-SLAM2 performs a **full bundle adjustment**, computing an optimal and consistent environment reconstruction.

ORB-SLAM2 has an embedded place recognition module based on *bag-of-words*, which reinitializes if the robot is in a known mapped scene. It also allows for relocalization if there is a tracking failure and loop detection. A net of neighboring keyframes created by the module enables the mapping and tracking to operate locally [2, 15]. Figure 2.6 depicts a visualization of ORB-SLAM2 during execution.

ORB-SLAM2 has many perks over its predecessor. The ORB features are robust to rotation and scale, present a good invariance to camera auto-gain and auto-exposure, and respond to illumination changes [2]. ORB-SLAM2's proposed localization capability allows zero-drift [3, 43] and lightweight localization for known environments. As it is an open-source algorithm, it provides versatility to enhancements and additions such as neural networks [44], autonomous navigation [15], or sensor fusion. Finally,

[4] A type of pose-graph optimization.

Figure 2.6: A visualization of ORB-SLAM2 during execution of a Stereo sequence. The green circles represent the ORB features that are being created and matched.

ORB-SLAM2 has a feature to save and load a map for localization. It also provides accurate trajectory and odometry estimations. This distance measurement is better when compared to RTAB-MAP [43].

2.2 Absolute Trajectory Error

Autonomous navigation is an important research topic for SLAM. There are several developed algorithms and practical implementations to approach the problem. Performance evaluation, that is the map quality and the robot's localization accuracy [45], becomes a critical point during the development and deployment. There is difficulty directly evaluating the map quality as there is a need to create a ground-truth map. Instead, a simplified evaluation process consists of analyzing the accuracy of the trajectory estimations [46, 47].

A benchmark and dataset[5] exist to address this performance evaluation problem. It has widespread use [3, 12, 20, 23, 48] for evaluating SLAM solutions and proposes two metrics: relative pose error (RPE) and absolute trajectory error (ATE) [46]. This thesis uses the latter for performance evaluation. ATE seeks the global consistency of an estimated course by comparing the available ground truth with the estimated trajectory. It evaluates the root mean square error (RMSE) over all the time indices

[5]These, together with executable scripts, can be found in TUM's Department of Informatics webpage at https://vision.in.tum.de/data/datasets/rgbd-dataset

of the translational components of a pose in each time instant (Equation (2.1)) [46].

$$RMSE(F_{1:n}) := (\frac{1}{n}\sum_{i=1}^{n} \parallel trans(\mathbf{F}_i) \parallel^2)^{\frac{1}{2}} \tag{2.1}$$

The rotational errors can typically manifest themselves in wrong translations. Therefore, the selected performance metric (Equation (2.1)) indirectly captures the rotational components. Furthermore, ATE permits visual inspection (Figure 2.7) due to its practical and intuitive visualization [49].

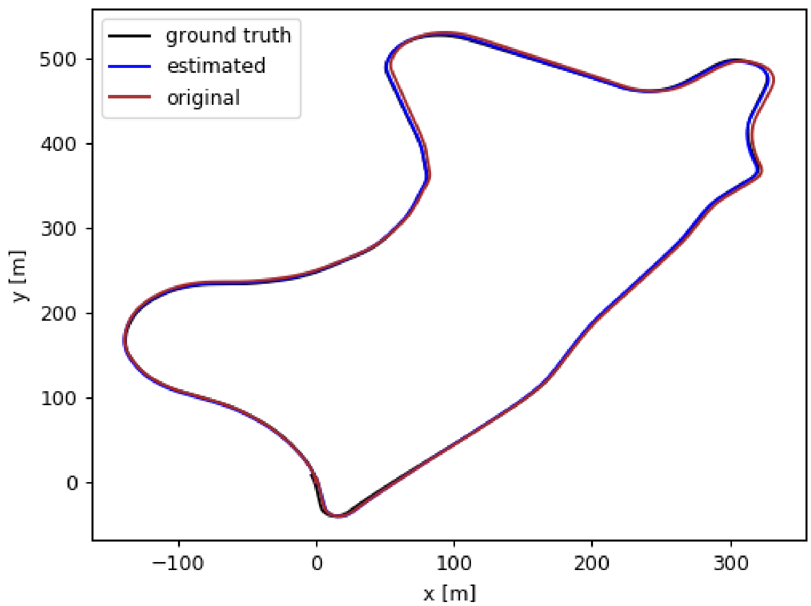


Figure 2.7: Visualization of the Absolute Trajectory Error (ATE) on the "KITTI sequence 09".

2.3 Hyperparameter Optimization

Hyperparameter optimization (HPO) aims to find the set of hyperparameters in a given model that returns the best performance when measured in a validation set [50]. To understand the optimization process, we must first differentiate parameters from hyperparameters. The former refers to the inputs that a model uses to make predictions, such as the weight coefficients in a regression model. Usually, model training learns these parameters [51, 52]. Hyperparameters solely depend upon the conduct of the algorithms when it is in the learning phase [52]. The user arbitrarily sets them before training the model [51]. In SLAM, they refer to the internal parameters of an algorithm, such as the number of RANSAC[6] iterations, the number of ORB features created for each image, or the minimum number of matched features to determine the culling of a keyframe. Essentially, parameters define how to use input data to get the desired output, and hyperparameters determine the structure of a model. For ORB-SLAM2's case, they affect the precision and functionality of each of its threads[7].

Hyperparameter Tuning (HPT) refers to the automatic optimization of the hyperparameters of a given model [53–55]. In addition to model learning, tuning is considered an extra step to find the set of hyperparameters that lead to the lowest error on a given validation set. This type of optimization results in a very costly function evaluation[8] and contains a generalized inaccuracy acquired through the validation [53]. Furthermore, a search space needs to be defined beforehand to apply HPT in SLAM[9].

HPO and HPT can be considered machine learning (ML) problems. Hyperparameters are important for ML since they directly control the behavior of training algorithms. HPO has a significant effect on the performance of machine learning models [57, 58]. Similarly, if a SLAM system is treated as a black-box function to optimize, it can use model-based or model-free algorithms to train it [59]. That is, HPO can modify the SLAM system's internal parameters to affect the computed ATE.

Model-based optimization algorithms construct a regression model[10] that predicts performance and optimizes the target function [60]. Contrarily, model-free optimiza-

[6]Random sample consensus is an outlier detection method for mathematical models.

[7]In Section 2.1.1, we defined them as tracking, mapping,loop-closing, and full-bundle adjustment.

[8]Due to the optimization entailing a model, the computation and performance evaluation can take hours or even days, which can be unscalable to more significant problems [56]

[9]This search space needs to set the limits for the hyperparameters and can benefit for using prior knowledge on the definition.

[10]Also known as a response or surrogate model.

tion algorithms do not use the model of the environment. Since the latter optimization approach relies on first-hand experience, the selected algorithm must search a parameter space to find a configuration that best minimizes the validation loss. Tuners can also use prior knowledge to delimit the parameter space and increase the efficiency of the obtained values.

Tuners often set default parameters with a general application context [54]. Case-specific parameter tuning will have increased performance when compared to the default settings. Empirically, having a set of fine-tuned parameters will minimize the generalized error. However, changing the default parameters often needs an expert with a good understanding of the system [61]. The cost of hyperparameter tuning increases exponentially as the number of parameters increases [50]. This exponential cost increment makes tuning time-consuming, which is its main disadvantage.

Several HPO algorithms are based on or borrow ideas from traditional optimization techniques and statistical model selection [62–66]. Figure 2.8 depicts a taxonomy for the hyperparameter optimization approaches. Five significant categories divide the algorithms based on the type of approach used:

- Brute-force approach
- Search-based approach
- Model-based approach
- Learning-based approach
- Population-based approach

2.4 Brute-Force Approach

Brute-force approaches are manual tuning strategies that rely on generating a set of parameter configurations based on the *design of experiments* (DOE) or *full factorial*

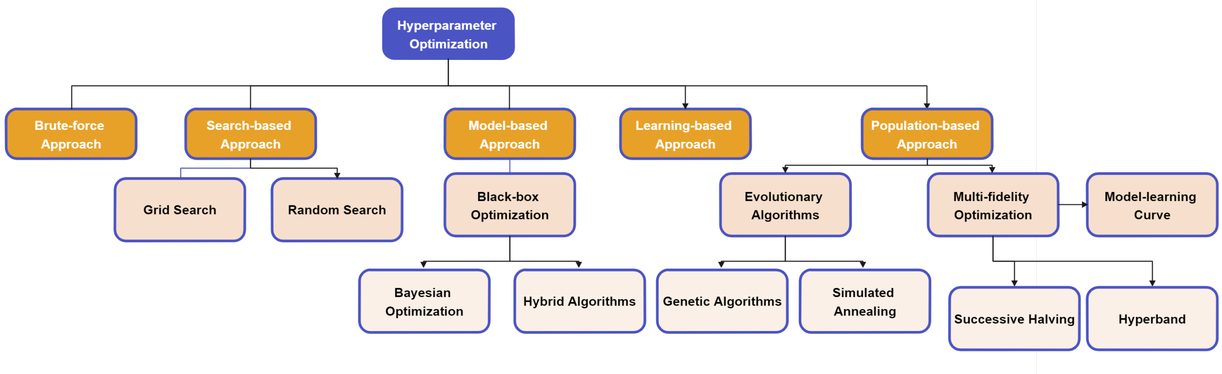


Figure 2.8: A taxonomy for the Hyperparameter Optimization techniques.

design (FFD) [54]. These candidate configurations are run the same number of times in each training instance[11]. Then, the best performance estimated is considered an optimal setting. This optimization approach is simple and easy to implement. However, the tuner must distribute the computational resources equally to all configurations [67, 68]. This distribution leads to a thorough exploration of poor-performing candidates. Furthermore, there are no existing criteria to determine the number of runs per configuration needed to handle the stochasticity derived from the target algorithm [54, 69]. Thus, manual tuning and brute-force approaches tend to be inefficient [70] and heavily biased by human actions and thoughts [71].

2.4.1 Brute-force approach in SLAM

Nevertheless, research shows that algorithm performance can increase via optimization if the parameter behavior is studied and there is a proper parameter space exploration and delimitation. For example, by tuning the selected parameters for SLAM, it is possible to reduce the navigation time of a Turtlebot [22]. Another case is the behavioral research of an odometer's parameters for underwater operation in Autonomous Underwater Vehicles (AUVs) [10]. They focused on two different odometers, with 10 and 16 parameters each, and identified the key parameters[12] to

[11] Based on the experiment design, the number of training instances may vary.
[12] Those are parameters that showed the most significant influence on the odometer.

reduce the optimization cost[13]. After brute-forcing an iteration over the determined parameter space, the study identifies the influence of each parameter on the search space concerning the translational and rotational errors. This research is relevant since there is a need for parameter selection to reduce optimization costs in cases where a large set of hyperparameters exist.

2.5 Search-based Approach

A type of approach for the optimization problem is search-based algorithms (e.g., grid search and random search). These are popular due to the algorithm simplicity, early-stopping capability [72], and easy implementation (Section 2.5.1 and Section 2.5.2). Search-based approaches (used as baselines in this thesis) are competitive due to their low computational costs and compatibility with SLAM.

2.5.1 Grid Search

Grid search is a simple basic solution for HPO. It consists of an exhaustive search and evaluation of all possible combinations within a parameter space [73–75]. Grid search can be computationally expensive as it may suffer from the *curse of dimensionality*[14] [1, 53, 74–76]. Figure 2.9 shows a representation of how grid search works. Despite the development of more specific and complex algorithms over the last decades, several reasons why it is still relevant as a state-of-the-art solution [77] exist:

- Simple implementation and trivial parallelization
- Global optimum convergence given enough time
- Reliability on low dimensional spaces (e.g., *1-D*, *2-D*)
- Low implementation complexity[15]

[13]A higher number of parameters increases the number of combinations exponentially, affecting computation costs.

[14]This means that the number of combinations grows exponentially with respect to the number of hyperparameters input.

[15]This means that the algorithm is easy to implement.

- Adaptability

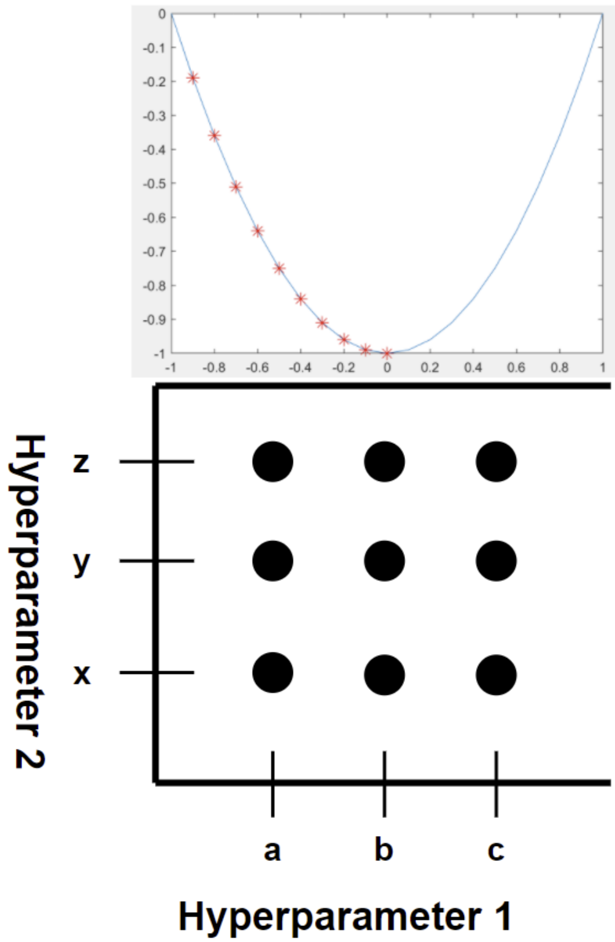


Figure 2.9: A visual representation of grid search's operation. Adapted from (Bergstra & Bengio, 2012)

2.5.2 Random Search

Random search is a variation of the previous algorithm. It randomly samples configurations in the parameter space instead of a Cartesian grid [53]. This algorithm requires that a budget be specified [1, 74, 75, 77]. The random search algorithm uses a given budget[16] constraint and tends to produce better solutions than grid search. Due

[16]Number of trials, time, etc.

to the parameter space exploration capability, random search increases the chances of converging into local optima [1, 70, 75, 78]. The main trade-off is computational effort [9]. Figure 2.10 depicts a general behavior of random search algorithms.

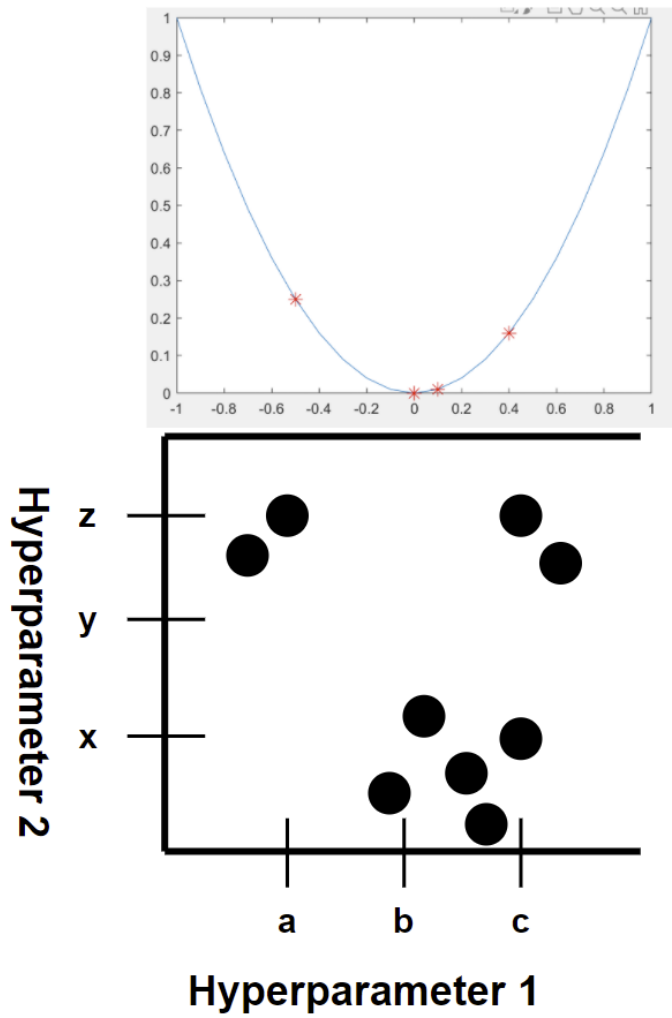


Figure 2.10: A visual representation of random search's operation. Adapted from (Bergstra & Bengio, 2012)

Random search algorithms are helpful for ill-structured global optimization problems, where the objective function may be non-convex, non-differentiable, and possibly discontinuous over a continuous, discrete, or mixed continuous-discrete domain [78]. This algorithm, like grid search, may suffer from the *curse of dimensionality*. Despite the limitation, it has many advantages that make it a relevant solution:

- Adaptability and customization to different cases (e.g., neural networks, SLAM, specific optimization problems, etc.) [79, 80]
- Simple implementation [70, 77, 78]
- Reliability in non-cubic parameter spaces [53]
- Trivial parallelization [53, 70, 77]
- Fast convergence to a local optimum [78]
- Asynchronous parallel execution [77]
- Early-stopping [77]

2.5.3 Search-based approaches in SLAM

Research has applied grid search to a feature-based SLAM solution to produce sets of hyperparameters that enhance trajectory estimation. The study focused on the visual odometry component and used ATE as the performance metric to determine the best-performing configuration [17]. One of the critical keynotes for tuning visual odometry in a feature-based SLAM system is the number of parameters to optimize. Additionally, studies used grid search to adjust model parameters for classification problems [73, 81]. As stated in previous sections, the computation cost increases exponentially as the number of combinations increases. Therefore, having a few optimizable parameters makes the problem computationally feasible.

Random search is another search-based approach used for optimization. Support Vector Machines (SVMs), like SLAM, use hyperparameters for their induced predictive models, affecting performance. A study suggests that random search can find a good-performing[17] set of parameters with similar results to grid search and meta-heuristics (e.g., GAs, Particle Swarm Optimization, and Estimation of Distribution),

[17]Good-performing is referred to as having a lower ATE value than the ATE produced by a default value execution.

but with a lower computational cost [82]. These optimization heuristics were tested on 70 different low-dimensionality datasets from the UCI[18] repository and proved that a simpler algorithm could provide competitive results.

However, both grid and random search possess some detrimental characteristics. They are heuristic, which ensues highly stochastic results. Furthermore, they have no defined execution time. Thus, they heavily rely on a termination criterion to avoid indefinite iterations. Additionally, these algorithms are fully explorative, having equally-distributed computational resources, which increase the execution time[19] [67, 68], and have a random convergence to a local optimum[20]. Therefore, search-based algorithms are simple strategies to implement with a SLAM solution but might not be optimal in obtaining good-performing configurations.

2.6 Model-based Approach

Model-based optimization approaches build a surface or surrogate model that describes the relation between parameter configurations and algorithm performance [54]. Then, they use these models to evaluate configuration candidates and guide the sampling process of new possible sets. That is, these models help find suitable parameter configurations for the target algorithm. These approaches favor complex optimization problems and clarify an algorithm's performance dependence on its parameter settings.

Model-based tuning algorithms use the obtained response models from their candidate configuration tests to provide desirable information to address the exploration-exploitation trade-off. These are sometimes considered an extension of the efficient global optimization (EGO) [83], which combines a predictive model with sequential sampling strategies such as the expected improvement criterion (EIC) [84] to identify

[18]Machine Learning repository at http://archive.ics.uci.edu/ml

[19]Meaning that time will be used in evaluating poor-performing configurations.

[20]This means that a good-performing solution can only be found after a certain amount of time and iterations have been done (i.e., this strategy finds a good-performing solution by chance).

promising good points [54]. Some of the most used model-based approaches include Sequential Kriging Optimization (SKO) [58, 85–87], Bayesian Optimization (BO) [21, 66, 88–90], and the state-of-the-art Bayesian Optimization Hyperband (BOHB) [91, 92].

2.6.1 Black-Box Optimization

Black-Box Optimization is the study and analysis of optimization problems and algorithms that assumes the objective function behaves like a black box [93]. A black box refers to an inaccessible function (i.e., no analytic description is available with observable outputs given some inputs [93])[21]. In HPT, this type of optimization refers to either blindly searching a parameter space or building an educated guess for a configuration that minimizes the validation loss [53].

Model-based approaches and many non-multi-fidelity optimization (Section 2.8.3) algorithms applied to SLAM for HPT do not exploit the detailed knowledge of the system but rather treat it as a black box [59, 86, 94]. This black box represents the state estimation results of vision navigation [28, 95, 96]. The objective of SLAM HPT is to optimize the results obtained from evaluating the map produced without considering the internal processes within the SLAM system.

2.6.2 Bayesian Optimization

Bayesian Optimization (BO) is a state-of-the-art algorithm that optimizes expensive objective functions [74, 75, 83, 90]. The central idea of this algorithm is to build a probabilistic model that can be updated and queried to drive optimization decisions, as shown in Figure 2.11 [66]. This approach is best applicable to non-convex problems with a non-closed-form expression for the objective function. BO produces noisy observations of the sampled values [88]. It is considered a highly effective optimization technique. This effectivity stems from its ability to incorporate prior knowledge

[21] We can define these functions as unknown or not having a closed-form solution.

about the problem to help direct the sampling, resulting in an exploration-exploitation trade-off in the search space [88]. BO takes its name from the *Bayes' Theorem*, which states that the *posterior* probability of a model, M, given some knowledge, E, is proportional to the *likelihood* of E given M multiplied by the prior probability of M (Equation (2.2)) [88]. The *prior* refers to the information that we know about the objective function. In contrast, the *posterior* is the captured information about the unknown objective function.

$$P(M|E) \propto P(E|M) \times P(M) \tag{2.2}$$

Algorithm 1 Bayesian Optimization

1: **for** $t = 1, 2, \dots$ **do**
2: Find $\mathbf{x}_t$ by optimizing the acquisition function over the GP: $\mathbf{x}_t = \text{argmax}_{\mathbf{x}}\, u(\mathbf{x}|\mathcal{D}_{1:t-1})$.
3: Sample the objective function: $y_t = f(\mathbf{x}_t) + \varepsilon_t$.
4: Augment the data $\mathcal{D}_{1:t} = \{\mathcal{D}_{1:t-1}, (\mathbf{x}_t, y_t)\}$ and update the GP.
5: **end for**

Figure 2.11: The Bayesian Optimization algorithm. Adapted from (Brochu, Cora, & De Freitas, 2010)

Bayesian optimization consists of mainly two components: the surrogate model and the acquisition function. The former component models the objective function, and the acquisition function measures the value generated by evaluating the objective function at a given point [1]. There are different options for surrogate models in BO: Random Forests, Gaussian Processes, or Tree-structured Parzen Estimators [1, 53]. Each has its advantages and disadvantages, but the standard for modeling the surrogate objective functions is Gaussian processes [1, 53]. When sampled on k random points, these are stochastic functions that follow a multivariate Gaussian distribution [53].

Gaussian process regression[22] is the process of adding additional information of the sampled points to the *prior* [53]. That is, fitting a response surface by a Gaussian

[22]This is also known as the concept of *kriging*.

process with a prior covariance. The mean provides an approximate response, but the predicted variance also produces valuable information. Then, a chosen acquisition function selects the following sample point within the parameter space [53]. Figure 2.12 depicts an example of the use of BO to find the parameters of a Support Vector Machine (SVM) classification that minimizes the cross-validated loss.

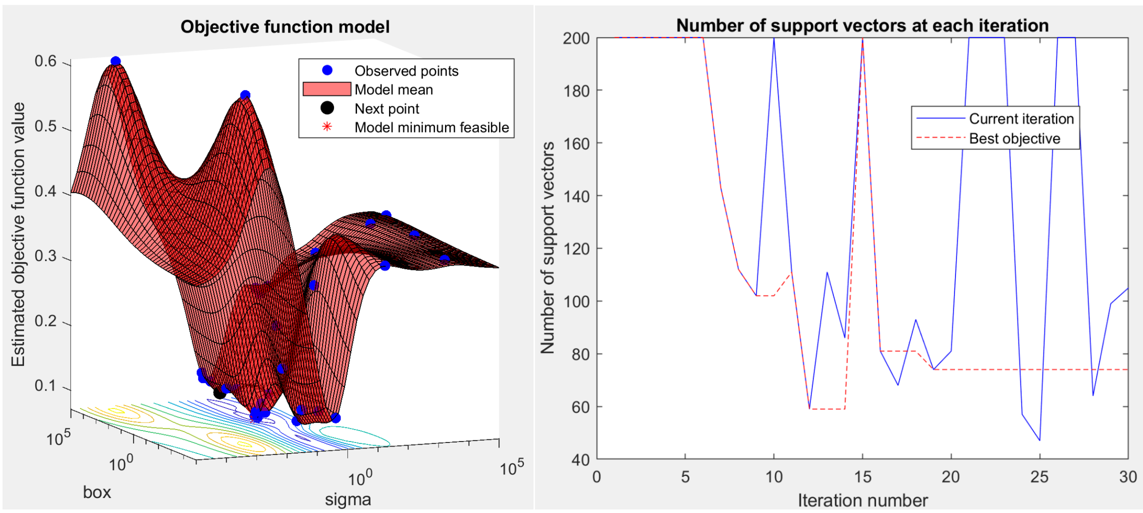


Figure 2.12: This is an example of Bayesian optimization that plots the number of support vectors as a function of the iteration number and graphs the number of support vectors for the best parameters found. Each blue dot represents an observed point obtained from optimizing the SVM. The black dot depicts the following observation in the model, and the red dot is the feasible model minimum. The red mesh represents the model mean.

BO is efficient for tuning a small number of hyperparameters. Its efficiency decreases as the search dimension increases[23] [53]. If the parameter space is too large, BO's behavior becomes par with random search [97]. One major drawback is the lack of parallelization of the algorithm compared to other baseline algorithms. The learning process needs to finish before a new one can be launched, as the Gaussian process and the acquisition function need to be updated to find a maximum [53]. Other drawbacks include the limitation on the types of hyperparameters that BO can work with (continuous and discrete, but not categorical). Computational costs and runtime can also be problematic. These can increase as the number of hyperparam-

[23] Bayesian optimization has a cubic complexity.

eters or the parameter search space increases. However, BO is a robust solution for finding a local optimum [92].

2.6.3 Hybrid Algorithms

More recent studies have shown that fusing optimization algorithms can increase performance, reduce execution time or obtain better results when compared to the uncombined components. These combinations may include members of the same optimization type (i.e., black-box optimization or multi-fidelity optimization) or a fusion of both. Since many combined optimization techniques exist, we will focus on the Bayesian-Evolutionary Algorithm (BEA) [98] and Bayesian Optimization Hyperband (BOHB) [92].

The Bayesian-Evolutionary Algorithm is an optimization algorithm that combines BO's data efficiency with the heuristic time-saving from an evolutionary algorithm (EA). It focuses on time efficiency, distinguishing BO or EA, and switching them accordingly. Figure 2.13 shows that the algorithm consists of three stages. The first[24] employs BO due to the low computation time [98]. When the time efficiency of the EA surpasses that of BO, the valuable knowledge transfers from the BO to the EA (i.e., the second stage). This stage employs the gains per time-interval approach to decide the switch point on when to transfer the information. In the last step, the search continues by an EA[25] with a hybrid adaptive and self-adaptive mutation strategy to balance exploration and exploitation [98]. Overall, BEA outperforms BO and EA regarding time efficiency and objective value [98].

On the other hand, BOHB (Figure 2.14) aims to combine BO and Hyperband (HB) to obtain strong anytime performance with a fast convergence rate to optimal configurations [75, 92]. Instead of using HB's blind repetition on top of successive halving (SH) [101], BOHB uses a BO approach [91]. BOHB relies on HB to determine how many configurations to evaluate within a budget. It replaces the random selection

[24]This stage refers to the early iterations of the algorithm.
[25]In principle any EA can be used for the third stage [99, 100]

Algorithm 1: BEA

Init: $\mathcal{N}_I$ initial samples $X_{i\in\{1:\mathcal{N}_I\}}(x_1, x_2, ..., x_n)$, i is iteration number; totally iterations $\mathcal{N}$; switch point iteration $\mathcal{N}_s$;
Result: solutions $X_{i\in\{1:\mathcal{N}\}}(x_1, x_2, ..., x_n)$ and f_i
while $i <= \mathcal{N}$ **do**
  if $i < \mathcal{N}_s$ **then**
    run Bayesian optimization ; ▷ *1st stage*
  else
    if $i = \mathcal{N}_s$ **then**
      transfer knowledge ; ▷ *2nd stage*
    run gain-aware EA ; ▷ *3th stage*

Figure 2.13: The BEA algorithm. Adapted from (Lan, Tomczak, Roijers, & Eiben, 2020)

of models at the beginning of each HB iteration with a model-based search [92]. The standard SH executes once it reaches the desired number of configurations at the beginning of each HB iteration.

Algorithm 2: Pseudocode for sampling in BOHB

input : observations D, fraction of random runs ρ, percentile q, number of samples N_s, minimum number of points N_{min} to build a model, and bandwidth factor b_w
output : next configuration to evaluate
if *rand()* $< \rho$ **then return** random configuration
$b = \arg\max\{D_b : |D_b| \geq N_{min} + 2\}$
if $b = \emptyset$ **then return** random configuration
fit KDEs according to Eqs. (2) and (3)
draw N_s samples according to $l'(\boldsymbol{x})$ (see text)
return sample with highest ratio $l(\boldsymbol{x})/g(\boldsymbol{x})$

Figure 2.14: The BOHB algorithm. Adapted from (Falkner, Klein, & Hutter, 2018)

Besides the combined advantages of BO and HB, BOHB also allows parallelization. It is an anytime algorithm that keeps track of the configuration that achieved the best validation performance; the algorithm can also be given a maximum budget of SH runs [92]. In summary, BOHB is an open-source[26], scalable, robust, and flexible

[26]The implementation can be found at https://github.com/automl/HpBandSter

algorithm that achieves both excellent final and anytime performances.

2.6.4 Model-based approaches in SLAM

The most popular algorithm used for VSLAM and multi-objective HPT is BO. One reason is that it provides an efficient solution to optimize the parameters for a visual-inertial SLAM system [48]. Furthermore, its compatibility with other algorithms (e.g., hybrid algorithms) positions model-based approaches relevant. One example is that a model-agnostic approach fused filter ensembles with BO for feature selection and fine-tuning the hyperparameters of an image classifier [102]. The main concern in VSLAM is the absence of a formal black-box function. Instead, we must consider the whole system for optimization, which leads to a problematic implementation of model-based algorithms.

Despite its heavy reliance on a fixed model, model-based approaches have many benefits as optimization algorithms (Section 2.3). They can adequate to different types of functions. Also, the surrogate model created for the target VSLAM system can produce highly efficient solutions.

2.7 Learning-based Approach

General performance and overall results select the default parameters in an established system. Tuning firmly assumes that a single set of parameters will work on average on every region of a complex problem [103]. Moreover, modifying the parameters requires an expert that has a keen understanding of the inner workings of the system used [61]. Because of these reasons and ease of use, *teleoperation* has been the solution for navigation and SLAM. Nevertheless, two learning-based approaches exist that erases the need for a human operator and make HPT feasible: reinforcement learning (RL) and learning from demonstration.

The former refers to learning a map from situations and actions to maximize a scalar reward for a given model [104, 105]. It consists of a learning agent that can sense the environment's state and take appropriate steps to maximize the rewards (instead of finding a generalized hidden structure from a function [106]) and the value function. Like BO [10, 53] and other model-based approaches, it focuses on the exploration-exploitation trade-off in search of the best optimization for a given function. The agent must try many actions and progressively favor those to appear to have the best rewards [106, 107]. Another feature is that the robot's navigation explicitly considers the whole problem of a goal-directed agent interacting with an uncertain environment [107].

Learning from demonstration alludes to seeking a good set of parameters that mimic the behavior of a teleoperated human demonstration of the desired navigation[27]. Because a human demonstration can behave differently at different points of the environment, a single set of parameters cannot closely resemble the execution in all its states. Thus, the problem divides into pieces that include consistent sensory observation and navigation commands, which lead to a relatively cohesive navigation behavior [61].

2.7.1 Learning-based approach in SLAM

It is common to pair learning-based, and reinforcement learning approaches. Both use the previously-obtained information to exploit the hyperparameter values and direct them towards better-performing configurations. Examples of this type of approach include using of an iterative Q-fitting algorithm to configure the parameters of a workstation to increase its capacity [108] and the application of RL for mapping in SLAM [105]. However, mimicking human operation is another way an algorithm can learn the best-suited parameters for a specific execution. APPLD[28] employs behavioral cloning to minimize the difference between the demonstrated actions and the actions that the robot would produce.

[27] This new set of parameters can be applied to a completely new environment by simply providing a teleoperated demonstration.

[28] Adaptive Planner Parameter Learning from Demonstration.

Learning-based approaches can lead to very positive results, but they are heavily biased. Since they need a human demonstration to start this process, an knowledgeable expert needs to set the initial example. Furthermore, the configuration combinations tend to become highly exploitative with no further perturbations to explore other solutions. This exploitation converges into a configuration that can produce better results for a specific environment. Learning-based approaches are suitable for adaptive parameter tuning but not for finding a set that shows an overall performance increase in the SLAM solution.

2.8 Population-based Approach

These approaches (Section 2.8.1) depend on particles or individuals to choose a suitable function to fit an environment [12, 109]. Each represents a potential solution (i.e., a data point), spread throughout the search space, to an optimization problem. The distribution of solutions finds the landscape of a problem [110].

Population-based algorithms (used in this thesis) are stochastic in nature. They favor an early exploration that becomes exploitative as more iterations occur. Hyperparameters in SLAM systems do not seem to have an established behavior (i.e., a change in one value in a configuration can produce unpredictable results). Hence the reason why population-based approaches are favorable for tuning is due to their fanning[29] and exploitation capabilities.

2.8.1 Evolutionary Algorithms

Evolutionary algorithms (EA) are a series of stochastic and heuristic optimization algorithms that mimic a natural biological behavior that follows the theory of evolution [16, 100, 110–113]. They share the same principle of incrementally improving the quality of a set of solutions over time [16, 114]. EAs rely on the concept of a *population* that undergo probabilistic operators such as mutations, recombination,

[29]The ability to spread out in the parameter space when defining the initial population.

and selection to evolve into better fitness values for individuals [100]. In other words, they follow the *Darwinist* theory of survival-of-the-fittest to explore the parameter space and converge to better fitness values by exploiting the most suitable traits. One significant property of EA heuristics is exploring the search space by a whole population of solutions. This search adds the flexibility of finding different local optima (in a process akin to random search) and resistance to premature convergence towards a single local optimum in multi-modal search spaces [112].

There are many different algorithms born from the concept of EA (Figure 2.15). Some are strict branching from the original evolutionary algorithm idea like genetic algorithms (GA) [75], genetic programming [113], evolutionary strategies [113], evolutionary programming [113], or learning classifier systems [112, 115]. Others are related search heuristics such as Tabu Search [113, 116, 117]. Most of these heuristics formulate from a concept found in nature, such as mimicking metallurgy through simulated annealing [65, 113], climbing a hill [118, 119], exploring an ant colony organization [113, 120], or the propagation and spread of birds in flight during migration [64, 75, 113]. In this thesis, we are primarily interested in GAs.

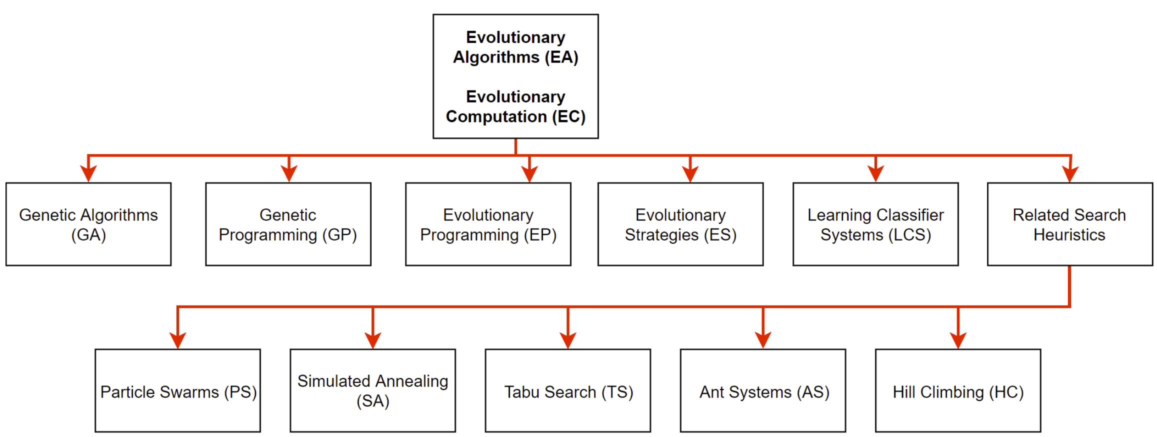


Figure 2.15: Classification and branching of Evolutionary Algorithm [112].

As their name suggests, genetic algorithms attempt to mimic the process of biological evolution and competition found in nature [62, 99, 121, 122]. The main idea of genetic-based optimization techniques is applying multiple genetic operations to a population of configurations (i.e., mutation, crossover) [1]. The algorithm consists of three main steps: selecting and pairing, mating, and mutating (Figure 2.16) [80, 114, 123]. These steps are then iterated over a set amount of time until they meet a termination criterion.

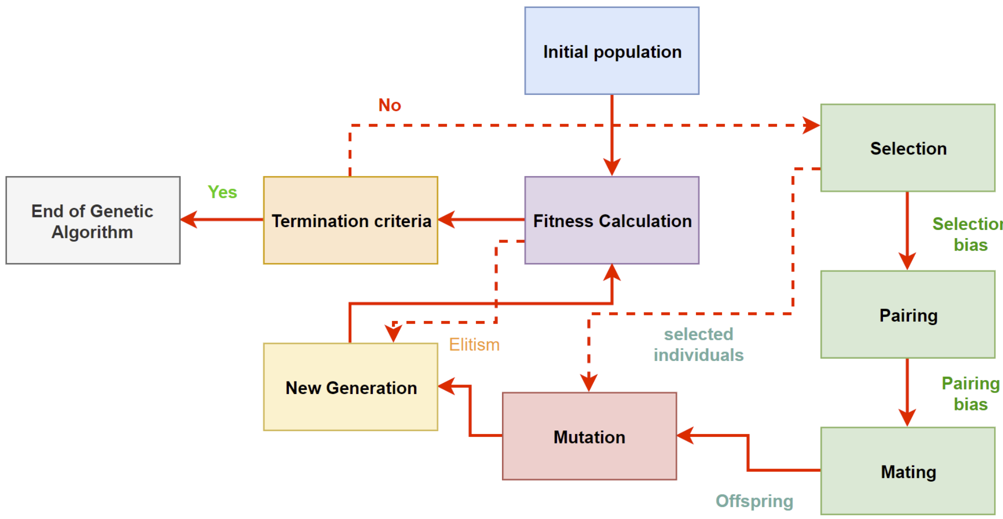


Figure 2.16: The Genetic Algorithm.

GAs follow the natural selection approach in biology, which means that the algorithm opts to eliminate the solutions with lower fitness values [99]. Selection and, subsequently, the pairing are the first vital operators in a GA. They refer to choosing a certain number of individuals within a population[30] and pairing them, using some bias (e.g., assigning weights depending on the fitness value [62], selecting the fittest results) as the basis for creating new individuals. The next step is mating[31]. Mating symbolizes the exchange of elements of each *parent* solution to produce two unique individuals that slightly differ from the originals. In theory, this swapping reflects the DNA combinations in sexual reproduction [99]. The last step is mutation. As the name suggests, this means that the algorithm chooses a random individual from the population and modifies its hyperparameter value(s) to create an entirely new

[30]It usually is the top half of the population, but the choice is dependent on the programmer.
[31]This step is also known as crossover.

solution[32]. Mutation helps the highly exploitative algorithm encourage exploration and escape local optima. Finally, this evaluation, reproduction, and mutation pattern repeats until a sufficiently satisfactory[33] solution appears to dominate the rest of the population [99].

GAs, and EAs in general, seem to perform well in a wide variety of optimization problems. Tuners consider them as general-purpose problem solvers [99]. Due to this, they tend to be outperformed by specialized algorithms. Nevertheless, GAs and EAs apply to spaces where no search heuristics are known [112]. These algorithms can be extended to complex problems [63, 112] (i.e., many-objective, distributed, and multi-objective algorithms, robotics) and are easy to implement [92], are adaptable [62], and can be applied to different hyperparameter tuning solutions [11, 12, 102, 104, 120, 121, 124].

Although GAs are considered simple, relevant state-of-the-art research still regards their practicality despite specialized algorithms. Its convenient exploitation of high-performing solutions combined with the stimulating exploration, given enough time, produces equally good results when compared with a model-based algorithm. They also exhibit compatibility with SLAM solutions. Hence, the use of these optimization algorithms for state-of-the-art SLAM HPO and HPT.

2.8.2 Simulated Annealing

Simulated annealing (SA) is a hyperparameter optimization algorithm inspired by metallurgy [65, 99, 100, 113], which simulates the heating and cooling of materials [100, 113, 125, 126]. It is a complex algorithm with several steps. It first selects a single starting value applied to all hyperparameters and evaluates the model performance [1, 65]. This initial parameter is supposed to be high enough to allow an aggressive random search over the parameter space [126]. Next, it will randomly update the value of one hyperparameter by selecting one contained within the neigh-

[32]Mutation is done rarely and just over a few individuals within the population.
[33]Low cost.

boring states. At first, it allows uphill movements readily, but as time passes, those perturbations tend to decrease until the hyperparameter value reaches its final form[34] [125, 126]. Finally, it compares the current model performance with that of the neighboring states [1]. The user then can reject or accept the neighboring state as the current one by using some deterministic criteria. Figure 2.17 shows a graphic representation of SA's behavior.

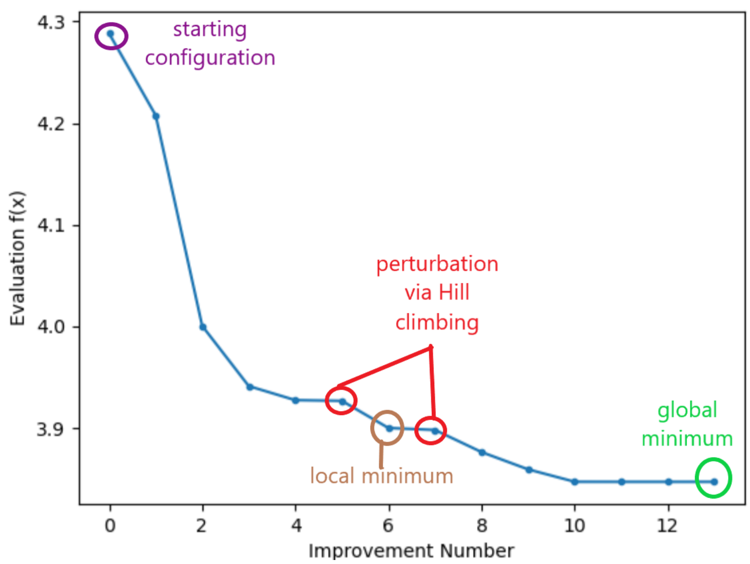


Figure 2.17: A graphic representation of the simulated annealing algorithm [125].

2.8.3 Multi-Fidelity Optimization

Multi-fidelity optimization refers to any optimization technique that focuses on decreasing the evaluation cost by combining many low-fidelity evaluations and a small number of expensive high-fidelity evaluations [1, 75]. This type of optimization is essential when working with large datasets. The training time, which can take days, can be reduced substantially by using cheap low-fidelity evaluations on a data subset. Although the high-fidelity evaluation can produce precise and accurate results for the whole dataset, the low-fidelity evaluations reduce the total evaluation cost. Thus,

[34] This type of algorithm closely resembles a standard downhill-only iterative improvement [89].

it can achieve a significant speed-up by trading the overall performance [1]. Usually, that increased approximation error can be negligible compared to the total time reduction achieved. We will present an overview of the most popular multi-fidelity optimization algorithms in the following subsections.

2.8.4 Model Learning Curve

Modeling a learning curve is an optimization technique that, during hyperparameter optimization, determines whether to allocate more resources or terminate the training for a particular configuration [1] given a learning curve. This algorithm may serve to model the performance of a given hyperparameter within a subset of a dataset. There are many ways to implement a termination procedure if the model performs poorly for a particular configuration. One method is learning curve extrapolation [1, 83], which terminates the execution if the performance of the predicted set is lower than that of the best model trained so far in the optimization process. Furthermore, a fusion of this optimization technique with other algorithms (e.g., Bayesian optimization) can reduce error and execution time models.

2.8.5 Successive Halving

Successive Halving (SH) is a bandit-based approach for optimization. It can provide accurate results within a relatively short execution time by allocating more resources to promising hyperparameter configurations [74, 75, 101, 127]. This algorithm (Figure 2.18) assigns a budget to evaluate all the hyperparameter sets. Next, they are ranked based on their performance [1]. Half of these configurations are culled based on their rank[35] values. Finally, the budget of the previous executions is doubled and repeated until one set remains [1, 54]. This algorithm outperforms the uniform budget allocation techniques regarding computation time and the number of iterations required [54]. However, it suffers from an exploration-exploitation stopgap. That

[35]The bottom half of the configurations are considered unnecessary and removed.

means that the user needs to determine if they need to allocate a large portion of the budget on exploring a vast number of configurations and a smaller budget tuning them or vice versa[36].

Successive Halving Algorithm

input: Budget B, n arms where $\ell_{i,k}$ denotes the kth loss from the ith arm

Initialize: $S_0 = [n]$.

For $k = 0, 1, \ldots, \lceil \log_2(n) \rceil - 1$

Pull each arm in S_k for $r_k = \lfloor \frac{B}{|S_k| \lceil \log_2(n) \rceil} \rfloor$ additional times and set $R_k = \sum_{j=0}^{k} r_j$.

Let σ_k be a bijection on S_k such that $\ell_{\sigma_k(1),R_k} \leq \ell_{\sigma_k(2),R_k} \leq \cdots \leq \ell_{\sigma_k(|S_k|),R_k}$

$S_{k+1} = \left\{ i \in S_k : \ell_{\sigma_k(i),R_k} \leq \ell_{\sigma_k(\lfloor |S_k|/2 \rfloor),R_k} \right\}$.

output : Singleton element of $S_{\lceil \log_2(n) \rceil}$

Figure 2.18: The Successive Halving algorithm. Adapted from (Jamieson & Talwalkar, Non-stochastic best arm identification and hyperparameter optimization, 2016)

2.8.6 Hyperband

Hyperband (HB) is a parameter-free, bandit-based optimization technique that maximizes exploration and optimizes the search space when selecting from randomly sampled configurations [1, 74, 75]. The algorithm consists of two components: a) the successive halving subroutine and b) the iteration over different models with the partitioned resource [128, 129]. Like SH, Hyperband requires a budget as an input. Then, it partitions the budget into several configurations and assigns a limited resource to each group of hyperparameters. That is, it frequently performs the SH algorithm with different budgets to find the best sets of hyperparameters [91]). Within each execution of the SH, a pruning factor (η) determines the number of sets to keep until it finds the best combinations of hyperparameters.

[36]Spending a small amount of the budget for exploration and the rest of it for tuning.

```
Algorithm 1: HYPERBAND algorithm for hyperparameter optimization.
input         : R, η (default η = 3)
initialization: s_max = ⌊log_η(R)⌋, B = (s_max + 1)R
for s ∈ {s_max, s_max − 1, ..., 0} do
    n = ⌈(B/R) · η^s/(s+1)⌉,      r = Rη^(−s)
    // begin SUCCESSIVEHALVING with (n,r) inner loop
    T = get_hyperparameter_configuration(n)
    for i ∈ {0, ..., s} do
        n_i = ⌊nη^(−i)⌋
        r_i = rη^i
        L = {run_then_return_val_loss(t, r_i) : t ∈ T}
        T = top_k(T, L, ⌊n_i/η⌋)
    end
end
return Configuration with the smallest intermediate loss seen so far.
```

Outer loop

Inner loop

Figure 2.19: The Hyperband Algorithm [97, 130].

In Figure 2.19, we can observe the HB algorithm. The blue section in the figure represents the integer number of hyperparameter sets considered at each loop iteration. The red portion refers to the rounded number of groups examined within the SH inner loop. Finally, the green part indicates the round number of configurations kept at the end of a given iteration [130]. In summary, HB is a robust, scalable, flexible, parameter-free, fast-converging, bandit-based algorithm that possesses the benefits from SH plus high implementation adaptability. It can also outperform BO given specific situations [98].

2.8.7 Population-based approaches in SLAM

Population-based algorithms have been used for years in VO and VSLAM to tune the hyperparameters, increase algorithm performance, and reduce an error metric. Particle Swarm Optimization (PSO) and GA were used to adjust the parameters of an RGB-D Visual Odometry solution [12]. Their results suggest that the procedure can replace baseline search-based approaches, execution speed is improved[37], and parameters can generalize to other sequences (while they share camera intrinsic values

[37]The execution time dropped from days to hours [12].

and a similar execution environment) [12].

Similarly, a study coupled a GA with a Lidar-Monocular (LIMO) VO solution [23] for parameter optimization. Their algorithm[38] (Figure 2.20) was run on the KITTI sequences and demonstrated that they could achieve better performance and a reduced translational error [46] across various tests [23]. Genetic Algorithms are also used for multi-objective solutions (MOGA), for example optimizing ROS[39] packages' parameters to enhance RTAB-MAP [121]. However, studies found that fine-tuned parameters may not outperform default value executions in all scenarios. This underperformance is critical for SLAM optimization. The objective is to find a set of fine-tuned hyperparameters that generally fit nicely on an anytime execution.

Like model-based approaches, tuners can combine population-based algorithms with other optimization methods or algorithms. A study combined a filter ensemble with a GA for image feature selection [102]. Another research coupled an EA with an adaptive constraint-handling technique for constrained optimization problems [104]. Furthermore, an EA applied to deep neural networks (DNN) can increase their performance [124]. All three previous cases demonstrate that population-based approaches are flexible, combinable, easy to implement, and produce good results when applied correctly.

2.9 Chapter Summary

Localization is estimating the robot's position within a known map [131] (Section 2.1). ORB-SLAM2 (used in this thesis) is a state-of-the-art solution for the SLAM problem (Section 2.1.1). HPO is the problem of choosing a set of optimal hyperparameters for a learning algorithm. These algorithms divide into five categories (Section 2.3): brute-force (Section 2.4), search-based (Section 2.5), model-based (Section 2.6), learning-based (Section 2.7), and population-based (Section 2.8).

[38] It is an open-source algorithm found at https://github.com/aralab-unr/LIMOWithGA
[39] Robot Operating System. A software found at https://www.ros.org/

Algorithm 1 GA-LIMO

Choose population of n chromosomes
Set the values of parameters into the chromosome
Run LIMO with the GA selected parameter values
for all chromosome values **do**
    Run LIMO on KITTI odometry data set sequence 01
    Compare LIMO estimated poses with ground truth
    Translation error σ_1 is found
    Run LIMO on KITTI odometry data set sequence 04
    Compare LIMO estimated poses with ground truth
    Translation error σ_4 is found
    Average error $\sigma_{avg} = \frac{\sigma_1+\sigma_4}{2}$
    return $1/\sigma_{avg}$
end for
Perform Uniform Crossover
Perform Flip Mutation at rate 0.1
Repeat for required number of generations for optimal solution

Figure 2.20: Algorithm that shows the combination of a GA with the LIMO VO system. Adapted from (Sehgal, Singandhupe, La, Tavakkoli, & Louis, 2019)

HPO can apply to different SLAM algorithms if treated as a black-box function (Section 2.6.1) in a machine learning problem. Among the different approaches, population-based algorithms seem to have a fair implementation for SLAM optimization solutions. This thesis uses model-free, population-based algorithms to train ORB-SLAM2 to produce a set of hyperparameters with lower error output than a default configuration execution. We use the proposed metric (absolute trajectory error) to evaluate the performance of the SLAM optimization.

Chapter 3

Methodology

This chapter discusses the proposed methodology for SLAM optimization. Section 3.1 explains the environmental setup used and the constraints considered for the experiments. Section 3.2 focuses on ORB-SLAM2's parameters, defining a parameter space and determining the influence of said parameters on the SLAM solution. Section 3.3 and Section 3.4 cover the chosen training and testing data sequences and the model-free algorithms chosen for optimizing SLAM.

3.1 Environmental Setup and Constraints

This section defines the equipment and environment used to execute the experiments found in Chapter 3 and Chapter 4. It details the modifications done to the ORB-SLAM2's source code to implement optimization approaches. Furthermore, it specifies the script used for performance evaluation and the constraints considered for the experiments.

3.1.1 Environmental Setup

One of the research objectives is the ease of reproduction of all simulations and experiments. We selected a Dell Latitude E5570 with an Intel Core i7-6820HQ CPU @ 2.70 GHz x 8 processors, 16GB RAM, and a 250 GB HDD capacity to fulfill that purpose (Figure 3.1). Additionally, we installed an Ubuntu 18.04.5 LTS[1] Operating

[1] Downloaded from https://releases.ubuntu.com/18.04/

System (OS) due to its stability and compatibility with ORB-SLAM2 simulations.

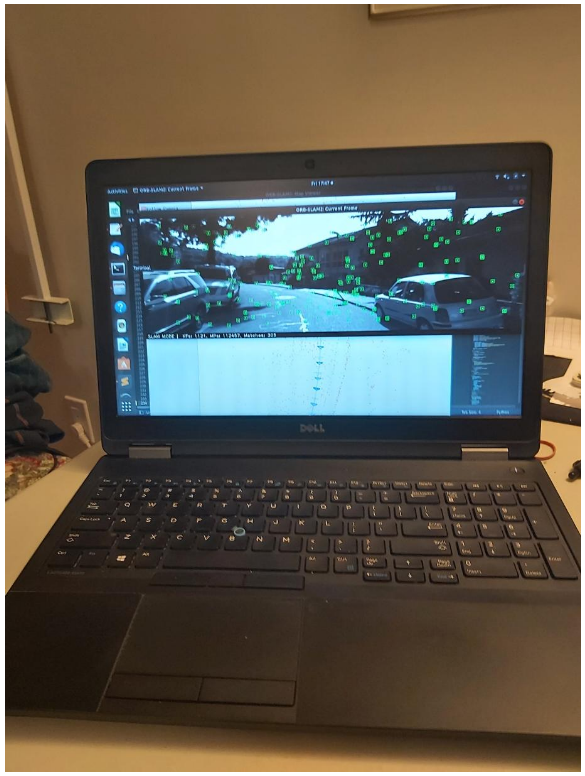


Figure 3.1: Dell Latitude E5570 used as setup and running an ORB-SLAM2 simulation

3.1.2 Modified SLAM System

Typically, a SLAM solution's source code contains hard-coded[2] values. It is necessary to alter the system's source code to create a modified version that accepts these fixed values as external inputs through a file. The modification of the SLAM system allows for the implementation of optimization algorithms without affecting the code itself. The optimization process can iterate throughout the external configuration file that the system reads. Then the process modifies the target parameters without affecting the functionality.

In this thesis, we altered the *.cpp*[3] files of ORB-SLAM2 to remove these fixed numbers. This way, we created an adaptable version of ORB-SLAM2. The hard-coded

[2]Fixed data that cannot be altered without modifying the program.

[3]These are the files produced by programming in a C++ environment.

data becomes external parameters read from the specific *.yaml*[4] file ORB-SLAM2 uses during execution. These adjusted files can accommodate the new variables[5], be modified and later optimized. The new parameters, the VO settings, and camera intrinsics[6] form an extensive repertoire of variables that control the behavior of the solution.

Similarly, other SLAM solutions might suffer from the same programmer's approach (hard-coded values). Therefore, a modification to their source code is necessary for optimization purposes. Removing the hard-coded values is not only a good programming practice but is also imperative for analyzing the influence of the variables on the SLAM solution.

3.1.3 Performance Metric Evaluation

Chapter 2 discussed ATE as one of the performance metrics [46] used for evaluating SLAM's performance. SLAM optimization must compare the trajectory estimated and the ground truth data to calculate the drift and error in the maps. We propose the usage of an available script[7] for the evaluation of SLAM HPO.

The *Evaluate ATE Scale*[8] script is modified from the RGB-D benchmark[9] to measure the Absolute Trajectory RMSE for the TUM [49] RGB-D dataset using ORB-SLAM2. Incidentally, this script is not compatible with the KITTI dataset. The ground truth format for the KITTI benchmark is different from the quaternion (TUM) format generated by the monocular ORB poses[10]. We use a modified version of the *Evaluate ATE Scale* script [132] to evaluate the ATE obtained from the experiments in Chapter 4.

[4] A type of text file, which ORB-SLAM2 uses to read all parametric values used in its execution.
[5] This modified version can be found at https://github.com/eimontecast/UOFA_Master_Thesis
[6] These should never be modified unless the camera settings change.
[7] Evaluate ATE
[8] Downloaded from https://github.com/raulmur/evaluate_ate_scale
[9] Available at https://vision.in.tum.de/data/datasets/rgbd-dataset/tools
[10] As seen on this issue https://github.com/raulmur/evaluate_ate_scale/issues/1

3.1.4 Constraints

Optimizing a SLAM solution is not simple. Several factors may hinder proper execution. These are some of the constraints considered during the parameter tuning process.

- HPO treats the behavior of SLAM as a black-box function [11, 121].
- The optimization process is to be considered a one-armed bandit problem[11].
- There is a consideration for the stochasticity of the ATE values produced by the performance evaluation script [132] during evaluation.

3.2 Parameter Selection

Chapter 2 defined the optimization process as an NP-hard problem[12] [133]. Section 2.5.1 disclosed that as the number of parameters increases, the complexity and computational costs for optimization also increase [1, 53, 76]. This section identifies the underlining parameters within a SLAM algorithm's source code, chooses an adequate parameter space to explore, and determines which variables influence the system the most. This research focuses on ORB-SLAM2 as our optimization target.

3.2.1 ORB-SLAM2's Parameters

ORB-SLAM2, similar to other SLAM solutions, has a certain number of parameters dedicated to specific functions. A spreadsheet[13] that identifies ORB-SLAM2's parameters was used and modified to include an additional number of hard-coded values found in the source code[14] (Appendix A)[15].

[11]It is a statistical learning model to make a sequential choice between several actions based on the rewards they generate.

[12]If anyone can translate an algorithm into one for solving any non-deterministic polynomial-time problem, it is NP-hard.

[13]This file was created by Sean Scheideman [45].

[14]A total of 112 values were hard-coded.

[15]It can be downloaded from https://github.com/eimontecast/UOFA_Master_Thesis

Table 3.1: ORB-SLAM2's parameters separated by role

Parameter Role	**Number of Parameters**
Intrinsic Camera Parameters	14
Viewer Parameters	7
ORB Parameters	5
Tracking	62
Loop Closing and Optimizing	23
Local Mapping	21
Miscellaneous	6
Total	**138**

Table 3.1 depicts the number of parameters considered for optimization and their specific role on the SLAM solution. The roles shown divide the parameters by the type of function regulated: viewer[16], loop closing and optimizing, local mapping, visual odometry (ORB-related parameters)[17], camera intrinsics[18], tracking, and miscellaneous[19]. From the 138 parameters found either within the source code or the *.yaml* files, only 117 are optimizable[20].

3.2.2 Computational Cost Reduction

The number of parameters optimized is directly proportional to the computational complexity for a given SLAM solution. Several studies on HPO approaches for SLAM dictate that only a few parameters are optimized [10, 17, 22, 23, 121]. That is to keep the computational costs feasible. We followed the examples of research where HPO fine-tuned SLAM and selected five parameters to optimize.

Parameter space size also affects the computational cost and complexity of opti-

[16]These are related to the camera viewpoints, the keyframe specifications, and linked to the camera's position.

[17]These link to the Visual Odometry portion of the SLAM solution.

[18]These parameters link to the camera's specifications, calibration and distortion parameters

[19]Parameters that could not fit in any other category.

[20]Both the intrinsic parameters and viewer parameters are immutable and unaltered for the sake of the algorithm's correct execution.

mization. A way to manage the *curse of dimensionality* is to limit the search space [17, 134]. A small number of parameters with a well-defined search space mitigates the exponential increase in complexity. It also reduces the runtime of algorithm training[21].

$$P_s(min) = \frac{\lambda}{2} \tag{3.1}$$

$$P_s(max) = 2\lambda \tag{3.2}$$

We delimited the search space for the experiments by using the default values as a starting point [17, 135] (Equation (3.1) and Equation (3.2)). The idea behind parameter space selection is to explore the area surrounding the manually chosen optimal parameters. In the defined bounded search space, λ equals the default value given to the parameter. Then, we select a step size[22] to increase the number of parameter values within the search space. It results in 21 different value options per parameter. Combining the parameter search space for the given number of chosen parameters, we obtain 21^5 possible combinations[23] for the parameter space.

3.2.3 Parameter Influence

Section 3.2.2 discussed the need for a well-defined parameter search space and a small number of parameters to reduce the computational costs. Now that the number of parameters is defined, we must choose which to optimize. One approach used in HPO for VO [10] is to use Spearman's correlation coefficient [137, 138]. It is a *distribution-free*[24], non-parametric approach of the Pearson correlation coefficient. It measures the monotonic[25] association between two variables based on their ranks [138].

[21] In case it is not delimited by a timer for each given execution.

[22] In HPT, the step size influences to what extent newly acquired information override old information, it metaphorically represents the speed at which a model learns [136].

[23] These are a total of 4,084,101 different parameter combinations.

[24] A distribution-free test does not assume the shape of the distribution for the drawn data samples [137].

[25] The variables also tend to change together in a monotonic relationship, but not necessarily at a constant rate.

Parameters in SLAM do not have a linear behavior. For example, a slight modification of a value would result in a different ATE result than the previous execution. Additionally, the loss of tracking in SLAM and the stochasticity of the results can result in significant outliers. We use Spearman's approach for SLAM because it fails to meet the underlying assumptions of Pearson's correlation [137, 138]:

- The data is not normally distributed or has a non-linear relationship.
- There is an ordinal value[26]. Not applied to SLAM, but is one of Pearson's correlation ordinances.
- The data exhibits significant outliers[27].

The objective of Spearman's correlation analysis is to identify[28] the most influential[29] parameters within an algorithm's structure (Figure 3.2). Sensitivity refers to the effect of subtle changes within a parameter's values on the algorithm's output. Namely, how much a difference in the parameter affects the error metric after an execution. The higher the sensitivity of the parameter is, the higher its effect on the ATE in ORB-SLAM2. That is, changes to the parameter values will significantly increase or decrease the error obtained.

The rank correlation coefficient obtained, represented by the letter ρ[30], measures the strength and direction of the relationship between the ranks [138]. It can take any value ranging from -1 to 1. The closer the absolute value of the coefficient is to 1, the stronger the relationship they possess:

- 1 represents a perfect correlation

[26]If values can be placed into first, second, or third order, then it is considered ordinal data.

[27]Unlike Pearson's correlation coefficient, Spearman's approach calculates the ranks. Thus, it is insensitive to outliers [138].

[28]Identifiability refers to the relationship between parameters [139]. A parameter is structurally identifiable if it estimates its value by observing the model output. Identifiability is not in the scope of this research.

[29]Also known as sensitivity.

[30]Also known as Spearman's ρ [138].

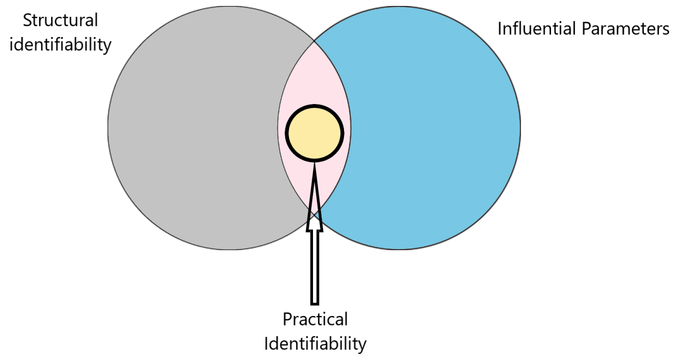


Figure 3.2: Relationship between influential, structural, and practical identifiable parameters [139].

- -1 signifies a negative correlation
- 0 means no correlation

Spearman's correlation formula changes depending on the existence of tied ranked values[31]. The application of Equation (3.3) happens when there are no draws within the ranks, where d_i represents the difference between ranks and n is the number of observations. Otherwise, Equation (3.4) implements a modified version of Pearson's approach, where $R(x)$ and $R(y)$ are the ranks of the x and y variables and $\overline{R(x)}$ and $\overline{R(y)}$ are the mean ranks.

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \tag{3.3}$$

$$\rho = \frac{\frac{1}{n}\sum_{n-1}^{n}(R(x_i) - \overline{R(x)}) \cdot (R(y_i) - \overline{R(y)})}{\sqrt{(\frac{1}{n}\sum_{n-1}^{n}(R(x_i) - \overline{R(x)})^2) \cdot (\frac{1}{n}\sum_{n-1}^{n}(R(y_i) - \overline{R(y)})^2)}} \tag{3.4}$$

3.2.4 Spearman's Correlation Calculation

We apply Spearman's correlation formula to analyze the relationship between a given parameter and the resulting ATE from running ORB-SLAM2. We executed the

[31] The same rank assigned to two or more observations.

SLAM solution once per value defined in the parameter space and considered the step size chosen in Section 3.2.2 for each candidate parameter (Section 3.2.1)[32]. Each individual run produces an ATE result, whose value can take any number within a range.

As mentioned in Section 3.1.3, consideration of the stochasticity in the results from an ORB-SLAM2 execution is a must. Thus, we repeated the previous implementation (running each parameter value once per parameter and recording their output ATE) five times. Then, we obtain an average for each group of resulting fitness values[33]. Table 3.2 displays an example of the variation of parameter values. It shows the ATE results and the computed mean for each case for the *ORBextractor.nFeatures* parameter.

The implementation of Spearman's correlation coefficient formula requires ranking the two sets (Parameter Values and Average ATE) and calculating the difference between ranks (defined as d). It defines ranking as assigning a numerical value to an individual compared to a list of other numeric values. If a list contains duplicated values, it gives an average to each set of duplicates[34]. Table 3.3 exhibits the ranking of each parameter value, average ATE, and the calculation of the difference between ranks (d) for the *ORBextractor.nFeatures* parameter.

Spearman's formula (Equation (3.5)) requires the summation of all the squared rank differences (e.g., the total calculated in Table 3.3). The total amount of elements evaluated (n) is substituted in the equation and used to calculate the resulting coefficient (Equation (3.6)). Equation (3.7) displays the result of calculating the coefficient for the *Orbextractor.nFeatures* example in Table 3.2 and Table 3.3.

$$\rho = 1 - \frac{6 \sum d_i^2}{n \cdot (n^2 - 1)} \tag{3.5}$$

[32]The list of parameters can be found in Appendix A.
[33]ATE
[34]See https://support.microsoft.com/en-us/office/rank-avg-function-bd406a6f-eb38-4d73-aa8e-6d1c3c72e83a

Table 3.2: ORBextractor.nFeatures ATE result changes by modifying the parameter value within the delimited parameter space and computing the average for each variation.

Parameter Value	ATE Result 1	ATE Result 2	ATE Result 3	ATE Result 4	ATE Result 5	Average ATE
1000	0.1685	0.2095	0.2026	0.1974	0.2067	0.1969
1100	0.1994	0.2158	0.1985	0.2106	0.2690	0.2186
1200	0.1787	0.1881	0.1800	0.2090	0.2019	0.1915
1300	0.1867	0.2017	0.2217	0.2177	0.2248	0.2105
1400	0.2067	0.2564	0.2183	0.2339	0.2393	0.2309
1500	0.1818	0.1929	0.2128	0.1764	0.2123	0.1952
1600	0.1784	0.1973	0.1934	0.1874	0.1867	0.1886
1700	0.1835	0.1661	0.1770	0.1693	0.1850	0.1762
1800	0.1642	0.1761	0.2115	0.1889	0.1818	0.1845
1900	0.1669	0.1807	0.1733	0.1851	0.2103	0.1832
2000	0.1882	0.1693	0.2149	0.2170	0.2102	0.1999
2100	0.1749	0.1604	0.1922	0.1939	0.2070	0.1857
2200	0.2607	0.2156	0.1854	0.1784	0.2374	0.2155
2300	0.2262	0.1955	0.1837	0.1968	0.1980	0.2001
2400	0.1963	0.2054	0.1836	0.1938	0.1855	0.1929
2500	0.2640	0.1404	0.1823	0.1822	0.1723	0.1882
2600	0.2203	0.1594	0.2291	0.2110	0.1863	0.2012
2700	0.3090	0.2686	0.3013	0.2554	0.2031	0.2675
2800	0.3166	0.3576	0.2994	0.2869	0.1975	0.2916
2900	0.2536	0.2625	0.2601	0.2705	0.2008	0.2495
3000	0.1998	0.1949	0.1666	0.1879	0.2174	0.1933

This table depicts the variations of only one parameter

Table 3.3: Calculation of ORBextractor.nFeature's parameter and average ATE ranks, and their respective difference between ranks

Parameter Value	Average ATE	Parameter Rank	ATE Rank	d*	d^2
1000	0.1969	21	11	10	100
1100	0.2186	20	5	15	225
1200	0.1915	19	15	4	16
1300	0.2105	18	7	11	121
1400	0.2309	17	4	13	169
1500	0.1952	16	12	4	16
1600	0.1886	15	16	-1	1
1700	0.1761	14	21	-7	49
1800	0.1845	13	19	-6	36
1900	0.1832	12	20	-8	64
2000	0.1999	11	10	1	1
2100	0.1856	10	18	-8	64
2200	0.2155	9	6	3	9
2300	0.2001	8	9	-1	1
2400	0.1929	7	14	-7	49
2500	0.1882	6	17	-11	121
2600	0.2012	5	8	-3	9
2700	0.2674	4	2	2	4
2800	0.2916	3	1	2	4
2900	0.2494	2	3	-1	1
3000	0.1933	1	13	-12	144
				Total d^2	1204

***d represents the difference between ranks**

$$\rho = 1 - \frac{6 \cdot 1204}{21 \cdot (21^2 - 1)} \tag{3.6}$$

$$\rho = 0.2181818 \tag{3.7}$$

Similar to the previous examples, these equations are executed once for every parameter within the list (Appendix A). The higher the resulting coefficient is, the more influence it has on the system. Section 3.2.5 discusses the results obtained from using the correlation formula.

3.2.5 Spearman's Correlation Results

Table 3.4 displays and ranks the top-performing results obtained from using Equation (3.5). We define top-performing as having a correlation value closer to a value of 1 (as specified in Section 3.2.3). This thesis assumes that all parameters are independent and exhibit no covariance with each other. Independence means that the top-most parameter in the table has the most significant influence on the output error metric. We selected the top five results from Table 3.4 as the optimization parameters.

We ran tests to verify the robustness to parameter change of ORB-SLAM2. We randomly modified[35] parameter values and executed SLAM. We discovered that the parameter *HISTO_LENGTH* was underperforming. This underperformance resulted from ORB-SLAM2 freezing and, subsequently, crashing when the parameter took specific values. Since there was no apparent behavior on which particular value range of *HISTO_LENGTH* would cause the system to be unresponsive, and *EraseObservation.minObservations* was the closest most influential parameter[36], we decided that the latter was to be optimized instead.

Therefore, the parameters chosen as optimization candidates for this research were:

[35] Similar to a random search algorithm.
[36] Assuming no covariance exists between parameters.

Table 3.4: ORB-SLAM2 parameters with the highest Spearman Correlation Coefficient

Position	Parameter Name*	Spearman Correlation Coefficient
1	ORBextractor.iniThFAST	0.7532
2	ORBextractor.nLevels	0.6493
3	GlobalBundleAdjustment.Iterations	0.5584
4	ReconstructH.minParallax	0.5194
5	OrbMatcher.HISTO_LENGTH**	0.5064
6	EraseObservation.minObservations	0.4696
7	SearchInNeighbours.nnSecondNeighbours	0.4675
8	Relocalization.initialMinInliers	0.4610
9	SearchForTriangulation.ThMultiplier	0.4441
10	PnPSolver.SetRansacParameters.epsilon	0.4090
11	OrbMatcher.TH_LOW	0.3415
12	Track.voMatchThres	0.3350
13	SearchByProjection.th	0.3314
14	DetectRelocalizationCandidates.minCommonMultiplier	0.3246
15	CreateNewMapPoints.nn	0.3233
16	OrbMatcher1.LowesRatio	0.2922
17	Initializer.FindFundamental.th	0.2883
18	OrbMatcher.CheckDistEpipolarLine	0.2649
19	GlobalBundleAdjustment.robustHuberKernelDelta	0.2428
20	LocalBundleAdjustment.chi2ErrorThStereo	0.2272

***Parameter names have been shortened to fit the table**

****This parameter had underperforming issues when optimized**

Value variations of HISTO_LENGTH caused ORB-SLAM2 to crash

- ORBextractor.iniThFAST
- ORBextractor.nLevels
- GlobalBundleAdjustment.Iterations
- ReconstructH.minParallax
- EraseObservation.minObservations

ORBextractor.iniThFAST refers to the images. The images divide into a grid, and at each cell, extract FAST, imposing a minimum response. First, it assesses the *iniThFast* parameter. Otherwise, it sets a lower value in the absence of image corners. *ORBextractor.nLevels* specifies the number of levels in the scale pyramid. *GlobalBundleAdjustment.Iterations* defines the number of iterations executed during global bundle adjustment. *ReconstructH.minParallax* designates the determination of the minimum amount of parallax[37] to accept homography[38]. Finally, *EraseObservation.minObservations* establishes the minimum number of observations needed to keep a map point.

3.3 Benchmark Selection

Section 3.1 discussed the hardware and software used for experimentation. This section presents the selected dataset used by the model-free algorithms. Ground truth data is essential for the evaluation of the effectiveness of training. In SLAM optimization (assuming a black box behavior), a dataset requires comparing the training output with the empirical trajectory evidence. Thus, we propose the usage of the KITTI benchmark for SLAM HPO.

[37] The effect whereby the position or direction of an object appears to differ when viewed from different positions, e.g., through the viewfinder and the lens of a camera.

[38] An invertible transformation from a projective space to itself that maps straight lines to straight lines.

ORB-SLAM2 has in-built stereo example modules that implement the KITTI [116], TUM [49], and EuroC [140] benchmarks. The KITTI dataset[39], a large-scale outdoor environment stereo dataset that includes translations, rotations, and loop-closures consist of 22 sequences, saved in lossless *PNG* format, and freely available ground truth data. Also, the benchmark has widespread use in state-of-the-art research [15, 17, 23, 43, 86, 94].

ORB-SLAM2 has one of the best SLAM solutions that present high accuracy for the KITTI benchmark using stereo cameras [2, 3]. Moreover, the dataset provides sequences exploring open areas and residential districts; the amount of ORB features found, and the resulting ATE varies because they depend on the test environment (e.g., landmarks, number of orb features). Additionally, training an algorithm in one sequence does not directly impact the test executions.

The KITTI benchmark contains 11 sequences that have ground truth trajectories available[40]. We also modified some of them to increase the testing of the candidate configurations (Table 3.5). We did not change[41] sequences 08 and 09. They are our control tests during experimentation. That is, we use those two sequences to analyze the fidelity of the ATE obtained from the configurations (Section 4.3).

Sequences 04 and 04m present a residential environment with open areas and tight quarters. They have an average duration of 39 and 21 seconds per execution, respectively (Table 3.5), and share the configuration (*.yaml*) file with a few other sequences in the benchmark (Table 3.6). The shared file means some tests have the same intrinsic camera parameters and can be grouped by the *.yaml* file when testing.

3.3.1 Confidence Interval

A confidence interval (CI) is a statistical measure to determine the probability that a parameter will fall between a set of values [141–143]. It provides a range of pop-

[39]Found at http://www.cvlibs.net/datasets/kitti/index.php
[40]Sequences 00-10
[41]Modified sequences in Table 3.5 have a letter m next to the sequence number.

Table 3.5: KITTI sequences, number of frames in the sequence and its runtime

Sequence Number	**Number of Image Frames**	**Length (min)**
00	4540	8.98
01	1100	2.22
02	4660	9.07
03	800	1.64
04	270	0.64
05	2760	5.66
06	1100	2.24
07	1100	2.27
08	4070	8.14
09	1590	3.20
10	1200	2.47
00m	3124	6.19
01m	945	1.91
02m	2414	4.72
03m	512	1.05
04m	149	0.35
05m	622	1.31
06m	842	1.72
07m	520	1.07
10m	1066	2.20

ulation values, bounded above and below the statistic's mean, in which a sample is consistent[42] at a given level of confidence[43] [141–145]. The CI provides information on sample variability (precision) and accuracy, or certainty that the randomly drawn element is within the actual population [141–145].

[42]The likelihood that the bound area contains an unknown population parameter.
[43]For CI, 95 % or 99 % of confidence are the most used levels.

Table 3.6: KITTI sequences and the *.yaml* file used in each configuration

Sequence Number	***.YAML* File Name**
00/00m	KITTI00-02.yaml
01/01m	KITTI00-02.yaml
02/02m	KITTI00-02.yaml
03/03m	KITTI03.yaml
04/04m	KITTI04-12.yaml
05/05m	KITTI04-12.yaml
06/06m	KITTI04-12.yaml
07/07m	KITTI04-12.yaml
08	KITTI04-12.yaml
09	KITTI04-12.yaml
10/10m	KITTI04-12.yaml

The CI provides more information than mere point estimation [141]. A confidence level can be established through the sample's mean and standard deviation while assuming a normal distribution as represented by a bell curve [141, 144, 145]. This level represents the probability that the sample's true mean is between the calculated upper (Equation (3.8)) and lower (Equation (3.9)) bounds [138, 142, 143]. For example, a confidence level of 95 % suggests that the average is located between the delimited area 95 % of the time [141].

$$U_b = \overline{X} + C_i \tag{3.8}$$

$$L_b = \overline{X} - C_i \tag{3.9}$$

The performance evaluation of the model-free algorithms used in the KITTI benchmark (Section 3.4) found in Chapter 4 (Section 4.2 and Section 4.3) focuses on using confidence intervals to determine the effectiveness of the HPO. That is, if the configuration candidate shares an approximate mean with the default sequence[44], then the optimization produces no overall enhancement for the given test. If the upper bound value of the candidate's ATE is smaller than the lower bound value of the default ATE, then a performance increment is present. Otherwise, the candidate is detrimental to the solution's performance for the given test.

3.4 Model-free Algorithms

Model-free approaches, generally, are those that do not store or use previously-obtained information [107, 146–148]. It is sometimes difficult to define the difference between model-based and model-free algorithms[45] [147] as there is a spectrum between them [147, 149]. The clear distinction (Figure 3.3) is that model-based algorithms build a model and plan the following action based on the environment's transition function [106, 107, 147, 150]. Model-free algorithms rely on first-hand experience to obtain a value function. That is, model-free algorithms use experienced information with minimal prospection[46] at decision time [147]. The effectiveness of an approach [151], either model-based or model-free, depends on the capacity to adapt to the function's behavior. In SLAM, an unknown number of local minima coupled with a non-linear behavior of the black box (i.e., the system itself) makes model-free algorithms easier and faster to implement than the model-based counterparts.

The effect on the output error given small perturbations or changes in the internal parameter values is unknown. Minor modifications in a parameter value (e.g., changing a set value from 2.15 to 2.16) can result in significant ATE outcome variations. Because there is a non-definitive, non-linear behavior of the SLAM output regarding parameter changes, we propose using model-free HPO algorithms, precisely a population-based approach.

The following sections describe the model-free algorithms, configuration, and run-

[44] The full sequence

[45] Model-free algorithms are those identified under a model-free approach classification.

[46] In psychology, the generation and evaluation of mental representations of possible futures.

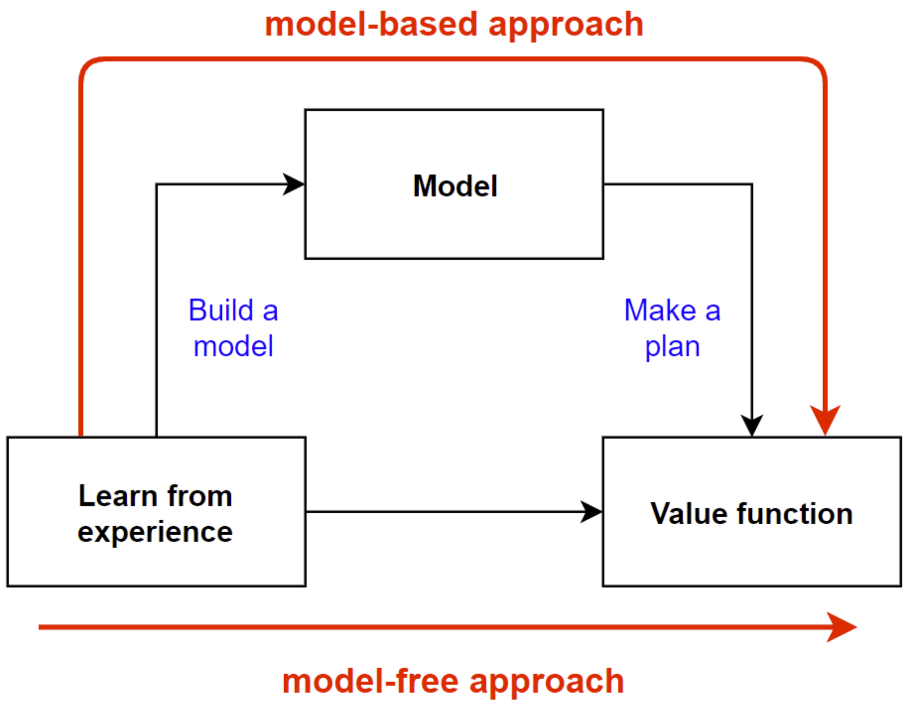


Figure 3.3: Model-based versus Model-free approaches [150].

time. Section 3.4.1 and Section 3.4.2 are the baseline algorithms chosen for evaluation and comparison with the selected optimization algorithms. Section 3.4.3 and Section 3.4.4 discuss the population-based algorithms (Genetic algorithm and Hyperband) chosen for SLAM optimization.

3.4.1 Grid Search

A baseline algorithm can produce the minimum expected performance on a given dataset [152]. Baselines are known for their simplicity and straightforward implementation. Section 2.5.1 defines grid search as an exhaustive search that trades execution time[47] to evaluate all the candidates in the parameter space thoroughly. State-of-the-art research widely uses this search-based algorithm. Thus, we selected it as one of our baseline model-free algorithms.

Typically, the computational budget [77] or the size of the parameter space deter-

[47] An exponential increase in execution time is dependent on the number of parameters and the size of the parameter space

mines the extent of a local search algorithm. Section 3.2.2 defines it as five parameters with 21 possible values; there are a total of 21^5 possible combinations within our search space. The chosen training sequence (KITTI 04), modified to have a reduced length execution, has a runtime of 0.35 minutes (Table 3.5). Equation (3.10) proves that a complete implementation of the grid search, even on the shortest testing sequence, is time-consuming.

$$time_{total} = 21^5 \cdot 0.35 = 2.72\, years \tag{3.10}$$

Nevertheless, local search heuristics can be subject to criteria to reduce runtime. A stopping rule, based on multi-start local search, can be applied to grid search if there is a moderately large number of local optima [153]. Additionally, a criterion could be used to an iterating search [154] to terminate if it meets a condition. Other options for minimizing time consumption are limiting the training times [155] or early termination [72]. In the latter case, the procedure can discontinue the experiments whose output is declining [72].

Algorithm 1 Grid Search Implementation

Require: Defined Search Space $[\frac{\lambda}{2}, 2\lambda]$, End Time
Ensure: *.Yaml* File has Default Values
  Create a Configuration Object
  while $(time_{current} \leqslant time_{end})$ **do**
    Run ORB-SLAM2
    Execute ATE Value Evaluation
    Append ATE to the Results Array
    Update Parameter Values
  end while
  Export Results to Excel

Algorithm 1 displays the implementation used for applying grid search to ORB-SLAM2. It requires having a defined parameter space and default values in the configuration (*.yaml*) file. We implemented an early-stopping strategy [72] since the ATE perturbations produced by the parameter changes are not predictable (Section 3.4), and the entire exploration is time-consuming. While the execution time elapsed is less than or equal to 14.21[48] hours, the program[49] executes SLAM, evaluates the ATE performance, and updates the parameter values for the subsequent execution. The outputs of the evaluation are exported into a readable format to facilitate analysis.

3.4.2 Random Search

Random Search is another simple algorithm used as a baseline for state-of-the-art research. Typically, it sacrifices a guarantee for optimality for swiftly finding a solution with convergence results in probability [78]. Several augmentations and modifications have enhanced the convergence time [153, 156–158]. It can achieve results faster than grid search but requires a stopping condition[50] [70, 77, 158].

Moreover, randomized algorithms can successfully obtain approximate optimal solutions [153, 159, 160]. That is, the randomization of parameter values extends the search space arbitrarily, allowing local optima to converge faster. Additionally, this heuristic has proven to obtain state-of-the-art efficiency [79, 97, 156] and competitive results on HPO [82, 161].

Algorithm 2 Random Search Implementation

Require: Defined Search Space $[\frac{\lambda}{2}, 2\lambda]$, End Time
Ensure: *.Yaml* File has Default Values
  Create a Configuration Object
  while $(time_{current} \leqslant time_{end})$ **do**
    Run ORB-SLAM2
    Execute ATE Value Evaluation
    Append ATE to the Results Array
    Randomly Change Parameter Values
  end while
  Export Results to Excel

Algorithm 2 is the random search implementation modified for its use in ORB-

[48] The time selected is the time taken for a complete execution of the Genetic Algorithm (Section 3.4.3).
[49] Coded in python and found at https://github.com/eimontecast/UOFA_Master_Thesis
[50] If no stopping condition is set, the algorithm will iterate infinitely.

SLAM2. Similar to Algorithm 1, it requires having a defined parameter space and a default configuration (*.yaml*) file. As long as the time elapsed has not surpassed the stopping condition[51], the program executes and evaluates the randomly produced configurations. Their values must be within the delimited search space. The resulting ATE outputs are then stored and exported to a readable file.

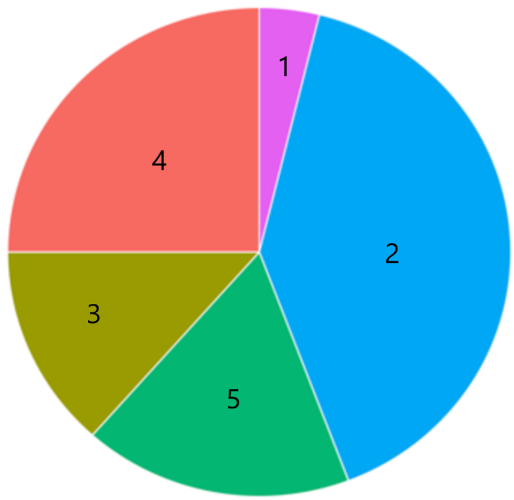
3.4.3 Genetic Algorithm

GAs are helpful for modeling solutions [75, 99] because they construct better strings or configurations from the best partial solutions of previous generations [11, 122]. These approaches make few assumptions about the underlying problem [11] and rely only on the consistency in the fitness function. The function acts as a measure of *goodness* to be maximized [123]. That is, individuals with a higher fitness value (lower ATE) are more likely to be selected as reproduction candidates.

Selection is a crucial step for a genetic algorithm. Some approaches for selection are roulette (Figure 3.4), fittest half, and random selection [62, 80, 123]. Roulette wheel selection can choose each individual based on the fitness value. Higher fitness values result in increased probabilities of being selected [80]. The fittest half approach sets the candidate solutions with higher fitness values, whereas the random method chooses arbitrary candidates for the next generation. We implemented the fittest half strategy on our GA.

Pairing and mating are a single operation in most genetic algorithm applications [80]. Pairing is the method where individuals chosen from the current population become the parents of the next generations [114]. There are many options for selecting individuals for crossover [80]: random, fittest, or weighted random. The first one refers to pairing two individuals arbitrarily. Fittest pairing is a method where individuals are paired, starting from the best individual [80]. It matches the most qualified individuals together, and the least suitable individuals are coupled to each other,

[51] 14.21 hours (Section 3.4.1)



Population	Fitness
1	5.0
2	40.0
3	20.0
4	25.0
5	30.0

Figure 3.4: A roulette-wheel marked for five individuals according to their fitness values. The fitter individuals have the higher chance of being selected.

which results in an easier culling. There are random sets of individuals in the weighted random selection, but fitter individuals are more likely to be paired [80]. Algorithm 3 uses the fittest pairing approach to ensure the exploitation of better candidates.

Reproduction or crossover is the operator that allows two chromosomes to exchange their genes for producing new offspring mutually [123]. Traditionally, the resulting offspring would substitute their parents if they proved to be fitter. Otherwise, the original individuals would survive in the population. *Bisexual reproduction* [62], also known as elitism [80, 114], was implemented because it resembles a natural process. The population consists of parents and offspring. When it reaches a maximum number, they compete for survival. Elitism ensures a natural evolution of the species [62] and diversity [114, 162]. In other words, a diverse population has a higher exploratory capability, which is essential at the start of the search process [114].

There are two options for gene crossover: single point and double point [80]. Single point selects a crossover point on the parent organism string and swaps all data, between the two parent organisms, beyond that point [80]. There is an exchange of genes between the two designated points in the double point crossover (Figure 3.5).

Both options result in the creation of two offspring. We implemented a double point crossover in Algorithm 3.

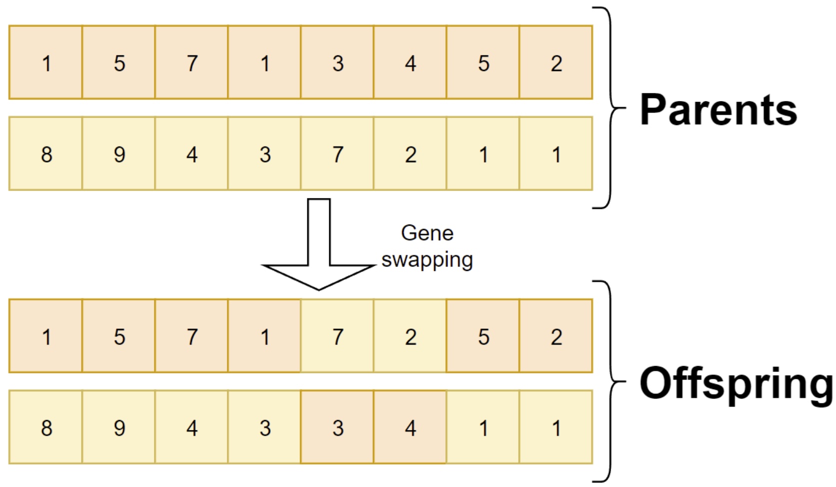


Figure 3.5: A double-point crossover overview.

The population size and the number of generations in a GA are vital factors to consider. The runtime needed for one generation of an EA is proportional to the population size [163]. If the population size exhausts the computational budget of the optimization process during the first generation, the algorithm will have a random sampling behavior [163, 164]. Larger population sizes have a minor effect on memory space than many generations [20, 103]. The bigger the population size, the greater the chance that the initial state of the population will contain a chromosome representing the optimal solution [20]; the number of generations also needed increases. Due to the size of our parameter search space and the number of parameters, we needed a relatively large population and few generations to reduce the computational cost and memory usage [20]. Similar to other research, we selected a maximum size of 200 individuals [165–169] and 15 generations for the implementation.

Algorithm 3 requires a defined parameter space and a default *.yaml* file. The implementation of the GA starts by creating an initial population of random configurations. Each configuration is set into the file, run in ORB-SLAM2, and evaluated.

Algorithm 3 Genetic Algorithm Implementation

Require: Defined Search Space $[\frac{\lambda}{2}, 2\lambda]$
Ensure: *.Yaml* File has Default Values
 $population = 200$, $generations_{max} = 15$
 Create Population of size *population* with Random Configurations
 for ($individual \in population$) **do**
  Execute ORB-SLAM2
  Evaluate ATE
 end for
 Sort Population by ATE
 Cull Half of the Population with Lowest ATE
 Pair Individuals with Highest ATE
 Reproduce
 Mutate 10% of the Total Population
 Export Generation Results to Excel File
 while ($generation \leq generations_{max}$) **do**
  Execute ORB-SLAM2
  Evaluate ATE
  Sort Population by ATE
  Pair Individuals with Highest ATE
  Reproduce
  Mutate 10% of the Total Population
  Sort Population by ATE
  Export Generation Results to Excel File
 end while

The pseudocode proceeds to sort the population by ATE performance[52] and remove the lower half. The process continues by pairing the fittest [80] potential mates and reproducing them using a double crossover approach [80, 129]. This approach ensures a broader parameter space exploration than a single crossover. A random mutation is applied to 10 % of the population to procure a better chance of escaping a local optimum. This process repeats until it reaches the maximum number of generations, where at every generational gap, it exports the results for a more straightforward analysis.

3.4.4 Hyperband

HB is a model and parameter-free algorithm chosen as a candidate for SLAM optimization. It has a fast runtime, and its results follow an exploitative and intuitive approach with a solid theoretical guarantee of correctness and sample complexity [97]. It prunes the configurations with higher ATE and re-evaluates the better-performing ones until a small number are left. A total of 3300 maximum epochs per configuration [97, 130] was defined with a pruning factor (η) of 3 [97, 126, 128, 130] to produce nine results after the last iteration of the algorithm.

Algorithm 4 displays the pseudocode for programming HB for SLAM. Similar to previous algorithms (Algorithm 1, Algorithm 2, and Algorithm 3), a defined parameter space and a default *.yaml* file are required. The program establishes the number of resources and the pruning factor (η) [130]. Then, it calculates the number of unique executions of SH (s_{max}) and the total number of iterations (without reuse) per execution of SH (*Budget*) [97]. It executes HB's finite horizon outer loop, where n is the number of configurations and r is the initial number of iterations run. After creating an initial population of random individuals, it executes the finite horizon of SH (n,r) for r_i iterations. The SLAM solution is run and evaluated. It averages the ATE obtained per configuration and keeps the best $\lfloor \frac{n_i}{\eta} \rfloor$ for the next iteration. In the

[52] Lowest ATE to highest

Algorithm 4 Hyperband Implementation

Require: Defined Search Space $[\frac{\lambda}{2}, 2\lambda]$
Ensure: *.Yaml* File has Default Values
  $resource = 3300$, $\eta = 30$
  Start Hyperband Algorithm
  $s_{max} = \lceil \log_\eta(resource) \rceil$
  $Budget = (s_{max} + 1) \cdot resource$
  for $s \in (s_{max}, s_{max} - 1, ..., 0)$ **do**
    $n = \lceil \frac{Budget}{resource \cdot \frac{\eta^s}{s+1}} \rceil$
    $r = resource \cdot \eta^{-s}$
    Create Population with Random Configurations
    for $i \in (0, ..., s)$ **do**
      $n_i = n \cdot eta^{-1}, r_i = r \cdot eta^i$
      while $(i \leq r_i)$ **do**
        Run ORB-SLAM2
        Calculate ATE
        Sum ATE to the variable $fitness$
        $i++$
      end while
      $fitness_{avg} = \frac{fitness}{r_i}$
      if $(i = s)$ **then**
        Save the $\lfloor \frac{n_i}{\eta} \rfloor$ Top Configurations
      end if
    end for
  end for
  Return the Top Configurations
  Export Results to Excel

end, we export the results to a spreadsheet for analysis

3.5 Chapter Summary

This chapter proposes a methodology for the implementation of HPO in SLAM.

3.5.1 Environmental Setup

The **preparation** of the SLAM system (environment) is necessary to implement an optimization algorithm. The process consists of modifying the source code of the SLAM solution to remove the hard-coded values. These become parameters inserted into a configuration file that the modified system accepts and reads during its execution. In the case of ORB-SLAM2, we added these values to the *.yaml* file that the system already uses as new parameters.

3.5.2 Parameter Selection

The **parameter selection** requires information on the influence of a given parameter on the SLAM system. Since SLAM can be subject to location tracking failure, the ATE results can become stochastic in nature, resulting in significant outliers. We propose using Spearman's correlation coefficient to estimate the effect of each parameter on the ATE produced. For ORB-SLAM2, the most influential parameters were calculated and tested in this section.

3.5.3 Benchmark Selection

The **benchmark selection** requires a dataset that has ground truth available. Comparing the output from running the benchmark on SLAM with the empirical evidence of the trajectory provides information on the accuracy and efficacy of a given configuration compared to a default run. We propose the usage of the KITTI benchmark for SLAM HPO.

3.5.4 Optimization Algorithms

The selection of **optimization algorithms** depends on the amount of information available for the optimization algorithm to use. The behavior of SLAM systems is complex and non-linear. That is, it is hard to make a prospection of the results of a given configuration. Therefore, model-free algorithms rely only on experience to obtain value functions, and are easier and faster to implement than model-based algorithms.

The optimization of a SLAM algorithm assumes that it is a black-box function. We propose using population-based algorithms (a genetic algorithm and Hyperband) to optimize SLAM's parameters. We will compare the results of these model-free approaches with those of two baseline search-based algorithms: grid and random search. The baseline algorithms have a time constraint linked to the runtime of the genetic algorithm to ensure a similar time execution.

The genetic algorithm has a reasonably large population of 200 individuals to induce exploration during its early stages. It is executed for 15 generations to achieve an optimal result (local minimum). We ran Hyperband with a resource[53] modification. It was changed to produce more results per execution of the algorithm.

[53]One of the two inputs needed for the algorithm.

Chapter 4

Experiments and Results

This chapter presents the results obtained from training and testing the model-free algorithms. Section 4.1 discusses the setup used for the experiments and the performance evaluation metric to determine the effectiveness of the configurations. We analyze the results obtained from training the algorithms in Section 4.2. Section 4.3 focuses on re-testing the resulting hyperparameter sets.

4.1 Experimental Setup

The KITTI benchmark was selected for training and testing the model-free algorithms. We shortened some sequences (Table 3.5) to increase the number of experiments done to the trained configurations. A modified version of sequence 04, $04m$, was used for training due to the decreased runtime per execution (0.35 min). We used the shortened sequences ($00m$ - $03m$, $05m$ - $07m$, & $10m$) to analyze the trained configurations' effectiveness. The remaining unchanged sequences[1] act as a secondary test to re-evaluate the promising configuration candidates that succeeded in reducing the SLAM runtime error in the previous trials. Since sequence 04 shares many similarities with its modified counterpart, we discarded it.

Section 3.4 discussed the algorithms selected for training our HPO approach for SLAM. We chose five influential parameters (Section 3.2.3) with a defined search space

[1]Sequences 00 - 03 and 05 - 10

(Section 3.2.2) for optimization. Each algorithm, coded in Python, had conditions and constraints applied for its execution (Section 3.4).

Table 4.1 displays the mean ATE obtained from executing each sequence 100 times with a default configuration. We used a confidence level of 95 % to calculate a confidence value that established the upper and lower ATE bounds for each sequence execution. We subjected the hyperparameter sets obtained from training to performance criteria (Section 4.1.1) to select candidates that outperformed a default execution.

We ran the trained configurations once on the shortened testing set. Section 4.1.2 provides the evaluation metrics used on the hyperparameter sets. Testing done on the unaltered sequences result in a group of elite candidates. We averaged 50 executions per configuration to obtain an ATE value with a smaller confidence window. Then, we apply the performance metrics (Section 4.1.2) to determine if a given configuration outperformed the default in all test cases.

The following subsections explain the performance evaluation metrics used in training and testing.

4.1.1 Training Evaluation

We evaluated the configurations obtained from training through a simple mathematical statement. Equation (4.1) defines the selection criteria for a configuration ($c_{selected}$) as a double conditional statement. That is, the ATE output of the trained configuration (c_{train}) is required to be below the lower bound threshold of the default execution[2] (Table 4.1) and not be equal to zero. The lower bound threshold (l_d) is the subtraction of the confidence value (C_d), calculated with a 95 % confidence level, from the mean ATE ($\overline{x_d}$). A value of zero would indicate an execution malfunction within ORB-SLAM2[3].

[2] A value of 0.1392

[3] A crash due to the configuration or initialization failure

Table 4.1: Default configuration mean ATE results and the upper and lower bound values, calculated with a 95 % confidence level, for 100 executions of each training and testing sequences

Sequence	Upper Bound ATE (l_d)	Mean ATE ($\overline{x_d}$)	Lower Bound ATE (u_d)
00	0.9659	0.9483	0.9307
01	6.7018	6.1590	5.6163
02	6.0005	5.8309	5.6614
03	0.2630	0.2556	0.2482
05	0.4126	0.3914	0.3702
06	0.6554	0.6134	0.5715
07	0.5025	0.4786	0.4548
08	3.4916	3.4030	3.3144
09	2.7480	2.4309	2.1137
10	1.0140	0.9724	0.9308
00m	0.9874	0.9665	0.9456
01m	6.1256	5.6833	5.2410
02m	6.2436	6.0591	5.8745
03m	0.2623	0.2581	0.2539
04m	0.2209	0.2131	0.2053
05m	0.4276	0.4186	0.4097
06m	0.6760	0.6372	0.5983
07m	0.4814	0.4703	0.4592
10m	1.0083	0.9576	0.9070

$$c_{selected} \implies [(c_{train} \leqslant l_d) \wedge (c_{train} \neq 0)] \ni l_d = \overline{x_d} - C_d \tag{4.1}$$

Equation (4.1) culls the configurations that do not show a performance increase or are underperforming[4]. The culling is a measure to prevent spending the computational budget on ineffective sets of hyperparameters. That is, the resources fur-

[4]Equations whose ATE value is higher than the lower bound threshold.

ther explore the better-performing configurations; this simulates SH's (Section 2.8.5) greedy approach [101].

4.1.2 Testing Evaluation

There are many evaluation metrics available for SLAM. Common quantitative ones are efficiency [170], average ranking [171], consistency [172], and accumulated error [2, 5, 172]. The success rate [173] is an evaluation metric that determines whether an objective function's result is equal or lower than the best possible value. Passing [13, 174, 175] and failure [13, 175] rates are performance metrics that determine the percentage of times a test succeeded or failed.

We modified the passing rate performance metric to fit our specific condition to determine if a tested configuration has a performance increment (an error reduction) [13, 174, 175]. The optimality rate (Δo) is the sum over all test cases where the configuration's upper bound ATE (s) is lower than the default error's lower bound value (l_d). The upper bound is the summation of the mean ATE and the confidence values[5]. This rate (Equation (4.2)) represents the percentage of successful tests with an increase in performance per configuration.

$$\Delta o = \frac{100}{s} \sum_{s \ni S} 1_{s<l_d \ni l_d = \overline{x_d} - C_d} \tag{4.2}$$

There are cases where the resulting ATE is neither optimal nor detrimental. The proximity rate (Δp) is the sum over all test cases where either the upper bound, lower bound, or mean ATE value of the tested configuration (s) is within the confidence interval of the default execution (Table 4.1). This performance metric is adapted from the success rate [173] only to consider equality, which means that the value is between the upper (u_d) and lower (l_d) ATE bounds of the default execution. Equation (4.3) represents the percentage of tests with neither an error reduction nor increment; the

[5]The mean and confidence value change depending on the configuration tested. The default values are static.

configuration is similar to a default execution.

$$\Delta p = \frac{100}{s} \sum_{s \ni S} 1_{(s > l_d \wedge s \leqslant u_d) \ni l_d = \overline{x_d} - C_d;\, u_d = \overline{x_d} + C_d} \tag{4.3}$$

The underperformance rate (Δu) is the opposite of the optimality rate. The base of this equation is the failure rate [13, 175]. It is the sum over all test cases where the configuration's lower bound ATE (s) is higher than the default error's upper bound value (u_d). Equation (4.4) displays the percentage of tests in which the optimization produced worse ATE results when compared to a default execution.

$$\Delta u = \frac{100}{s} \sum_{s \ni S} 1_{s > u_d \ni u_d = \overline{x_d} + C_d} \tag{4.4}$$

Equation (4.2), Equation (4.3), and Equation (4.4) provide helpful information about the configurations' performance across different scenarios. The proposed metrics offer more information than a simple pass[6]. They provide qualitative data on how the ATE obtained compares to the default error's confidence interval. In other words, they can determine if a configuration has better, worse, or overall similar results than a default execution.

Theoretically, a high optimality rate equates to showing a promising substitute for the default configuration. Nevertheless, that is not the case. We considered all three rates when analyzing the effectiveness of a hyperparameter set. For example, high rates in optimality and underperformance are not better than high rates in proximity with no underperformance. We will use these performance metrics to analyze the configuration candidates (Section 4.3).

4.2 Sequence Training Results

This section focuses on the products obtained from training the selected population-based and the baseline search-based algorithms. We organized the results into sub-

[6]Passing and failure rates

sections depending on the training algorithm used. Section 4.2.5 summarizes the findings from training the algorithms. It also discusses the possible causes that lead to the results.

4.2.1 Grid Search

We executed the grid search algorithm for a total of 14.21 hours. The exhaustive search of the parameter space produced 2163 results[7] over the defined time. We analyzed each candidate configuration before comparing the specified metric (Equation (4.1)). The analysis proved that a total of 1623 configurations were unusable. That is, the fitness metric provided was zero. The causes of this nil value can be plenty (e.g., initialization failure, tracking failure, program crash, script failure) but are not relevant for this research.

We applied the performance metric to the remaining 540 candidates (24.97 % of the population tested). The blue-colored rows, found in Appendix C, display the resulting configurations. The candidate culling resulted in 14.24 % (308) of the obtained initial hyperparameter sets showing an error reduction in ORB-SLAM2 (Appendix D).

4.2.2 Random Search

We used the random search algorithm to train SLAM similarly to grid search. We altered the Python implementation of the algorithm to only record the configurations whose ATE fulfilled the performance metric (Equation (4.1)). The aleatory sampling of the parameter space produced 105 results that matched the condition[8]. Since random search has the same execution time as grid search, we estimated 2163 configurations analyzed during the runtime.

We applied the performance metric to the 105 configurations (4.85 % of the estimated population tested). The gray-colored rows, found in Appendix C, display

[7] We explored a total of 0.002 % of the parameter space.

[8] The ATE needed to be a value other than zero and less than the default lower bound ATE.

the random sampling approach's results. We discovered that only 1.25 % (27) of the configurations increased the performance of ORB-SLAM2 for sequence $04m$.

4.2.3 Genetic Algorithm

The algorithm, initialized with a population of 200 and 15 generations (Section 3.4.3), was trained for a total runtime of 14.21 hours. The population was subjected to our performance metric (Equation (4.1)) to obtain 92.5 % (185) configurations that matched the stipulated condition. The green-colored rows, found in Appendix C, display the results from using this training algorithm.

We expected that the genetic algorithm produced a higher percentage of configurations that outperformed the default execution of sequence $04m$ than the other algorithms. GAs use a greedy approach to exploit the excellent parameter values and discard the underperforming ones. Hence, this model-free optimization approach is expected to surpass both the baseline algorithms and HB.

4.2.4 Hyperband

We set the resource and pruning factor parameters of HB to produce nine configurations per execution of the algorithm (Section 3.4.4). The runtime of this training algorithm was swift[9]. Hence, we executed HB five times to increase the number of candidates produced. We applied the performance metric (Equation (4.1)) to the 45 resulting configurations. 28.89 % (13) of the found configurations exhibited an error reduction. The yellow-colored rows, found in Appendix C, display the configurations and the fitness values calculated.

4.2.5 Configuration Candidates

Table 4.2 summarizes the number of configurations found per training algorithm (Appendix D). Grid search, Random Search, Genetic Algorithm, and Hyperband

[9] 2.71 hours

produced 308, 27, 185, and 13 candidates, respectively. This information indicates that a heuristic search has more results (given a temporal budget) than combined random sampling and greedy approaches. Nevertheless, exploitation-oriented algorithms might produce configurations that exhibit error reduction in more test cases[10].

Table 4.2: Number of configurations found by each training algorithm

Training Algorithm	**Number of Configurations**
Grid Search	308
Random Search	27
Genetic Algorithm	185
Hyperband	13
Total	**533**

There are many reasons why grid search might have found more configurations than the other approaches. The parameter space might have been too large for random sampling to find appropriate solutions given the time budget. Similarly, HB's design is such that it only produces a limited number of optimal hyperparameter sets. An increase in the number of resources or the pruning factor might result in a behavior akin to a random search. The population size of the GA limits the number of configurations it can produce. However, increasing the measure would require increasing the number of generations to converge to better-performing sets. Thus, increasing the computation costs.

4.3 Testing Results

This section discusses testing the trained configurations obtained from the model-free algorithms (Section 4.2). Section 4.3.1 evaluates the trained hyperparameter sets on modified[11] KITTI sequences. As mentioned in Section 3.3, we did not alter sequences

[10] This is similar to the quantity vs. quality problem.
[11] Shortened

08 and 09 to have a subset of trials shared in both test sets: a control group. It also adds a lax metric, success rate ($\Delta o + \Delta p$), to determine the number of candidates tested further. Section 4.3.2 assesses the collection of configurations that displayed the highest Δo[12] (Equation (4.2)) on the assortment of unaltered sequences.

4.3.1 Shortened Sequence Testing

We subjected the 533 configurations obtained through training the model-free algorithms to performance tests. That is, we analyzed the effectiveness[13] of the parameter values on different execution environments. We ran each set once for each unaltered sequence from the test set, in addition to sequences 08 and 09. Appendix E displays a table containing all the ATE values calculated for each parameter configuration per testing sequence.

We designated each candidate with a key name that indicates the training algorithm used to obtain the configuration. Section 3.1.3 proposed performance metrics to evaluate each contending configuration's efficacy. Appendix F displays the Δo, Δp, and Δu computed per configuration candidate.

Table 4.3 contains the highest-ranking contenders based on the optimality rate. That is, the hyperparameter sets that reduced the ATE of test sequences the most times. The top-ranked configurations had an optimality rate of 80 % or higher. Nevertheless, the number of candidates with a result of 90 % was meager. To increase the number of configurations to re-test, we introduced a metric, success rate [173] (defined as Δs in Equation (4.5)), that combines the rates of optimality and proximity ($\Delta o + \Delta p$). This metric indicates that a given configuration had a lower error than a default execution (Table 4.1) or remained within its confidence interval[14] (i.e., no increase or decrease in performance). This lax metric increased the number of selected configurations to test.

[12] Defined previously as optimality rate.
[13] Computed ATE
[14] Confidence intervals are always calculated with a 95% confidence level.

Table 4.3: Configuration candidates, tested on the modified sequences, that display the highest optimality rate

Configuration Name	Δo (%)	Δp (%)	Δu (%)	Δs (%)
Gen21	**90**	**0**	**10**	**90**
Grd80	**90**	**0**	**10**	**90**
Gen51	**80**	**10**	**10**	**90**
Gen68	**80**	**10**	**10**	**90**
Gen123	80	0	20	80
Gen160	80	0	20	80
Grd130	80	0	20	80
Grd192*	80	0	20	80
Grd213	80	0	20	80
Grd223	**80**	**10**	**10**	**90**
Grd224*	80	0	20	80
Grd234	80	0	20	80
Grd235*	80	0	20	80
Grd270	80	0	20	80
HB5	**80**	**10**	**10**	**90**

*Configuration randomly selected for further evaluation

$$\Delta s = \Delta o + \Delta p = \frac{100}{s} \sum_{s \,\exists\, S} 1_{s<l_d \,\ni\, l_d = \overline{x_d} - C_d} + \frac{100}{s} \sum_{s \,\exists\, S} 1_{(s>l_d \,\wedge\, s\leqslant u_d) \,\ni\, l_d = \overline{x_d} - C_d;\, u_d = \overline{x_d} + C_d} \tag{4.5}$$

Additionally, to further increase the number of configurations tested in Section 4.3.2, we randomly evaluated 30 % of the remaining top-performing candidates (Table 4.3). Adding new candidates increases the chances of finding a set of parameters that outperform the default execution on all test cases without raising the computation resources and memory of the hardware. It also serves to identify the difference gap in optimization performance between candidate[15] and selected[16] configurations. The parameters tested on the remaining unmodified sequences are:

- **Gen21** and **Grd80** due to the 90% of Δo achieved.
- **Gen51**, **Gen68**, **Grd223**, and **HB5**, because of the 90% success rate achieved.
- **Grd192**, **Grd224**, and **Grd235** as configurations with a Δo of 80% that were randomly selected from the options in Table 4.3

4.3.2 Full Sequence Testing

Due to their high optimality and proximity rates (Table 4.3), we subjected the candidate configurations to the second batch of performance trials on the unaltered sequences. We ran 50 times per test to calculate a mean ATE with a lower confidence interval. Sequences 08 and 09 were used as a control experiment to verify the results obtained from the sequence executions. Appendix E contains a table that includes the mean ATE computed for each configuration per test.

Similar to Section 4.3.1, the efficacy of the selected configurations was analyzed using the proposed performance metrics (Section 3.1.3). Additionally, we used the combined metric[17] defined (Table 4.3) to determine the success rate for each candidate. Table 4.4 displays the results of calculating the performance metrics for the sequence tests.

We can identify in Table 4.4 that the best-performing configurations from the previous tests[18] do not exhibit a high success rate. We notice that the randomly chosen promising[19] candidates display high rates of optimality and proximity. Additionally, they have a low underperformance rate. One of the leading causes for these results is the behavior of the confidence intervals of each configuration (Appendix H).

[15] High Δo
[16] High $\Delta o + \Delta p$
[17] $\Delta o + \Delta p$
[18] Gen21 and Grd80
[19] Configurations that had a success rate of 80 % on the modified sequence tests.

Table 4.4: Configuration candidates, tested on the unmodified sequences, that display the highest optimality rate

Configuration Name	Δo (%)	Δp (%)	Δu (%)	Δs (%)
Gen21	40	20	40	60
Grd80	30	30	40	60
Gen51	30	30	40	60
Gen68	20	50	30	70
Grd223	30	30	40	60
HB5	20	40	40	60
Grd192	40	30	30	70
Grd224	**40**	**40**	**20**	**80**
Grd235	**40**	**40**	**20**	**80**

Figure 4.1a exhibits one of the cases where the candidates failed to outperform the default execution; Figure 4.1b displays one case most of the candidates had very close proximity to the predetermined configuration.

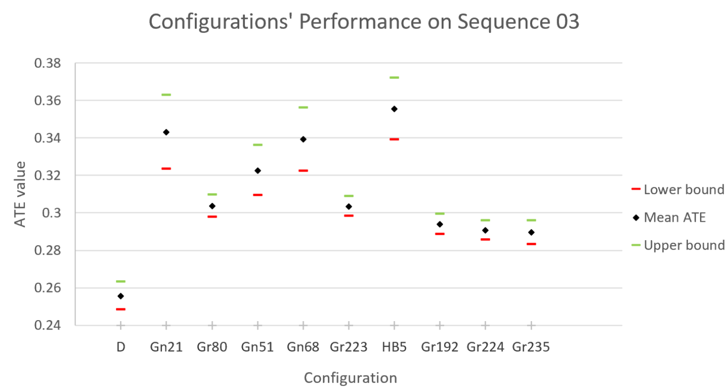
We theorize that the leading cause for the lackluster performance from the selected configurations could hint at overfitting the training sequence's ATE. Overfitting may be why most of the tests that share the same *.yaml* file (Table 3.6) with the training sequences showed either an error reduction or proximity with the default's confidence interval. Other possible causes for this behavior may be the execution environment, the number of ORB features detected, tracking failures, or the configurations that were not adequate for the testing cases. Nevertheless, Table 4.4 confirms that the maximum success rate achieved is 80 %.

Table 4.4 summarizes the information from Appendix G to facilitate the visualization of the performance obtained. Table G.1 displays the actual percentages in which each configuration's error changed concerning the default run. We gave a value of 0.00 to each case where the computed ATE was within the confidence interval of the default configuration. Even though the top-performing candidates (Grd224 and

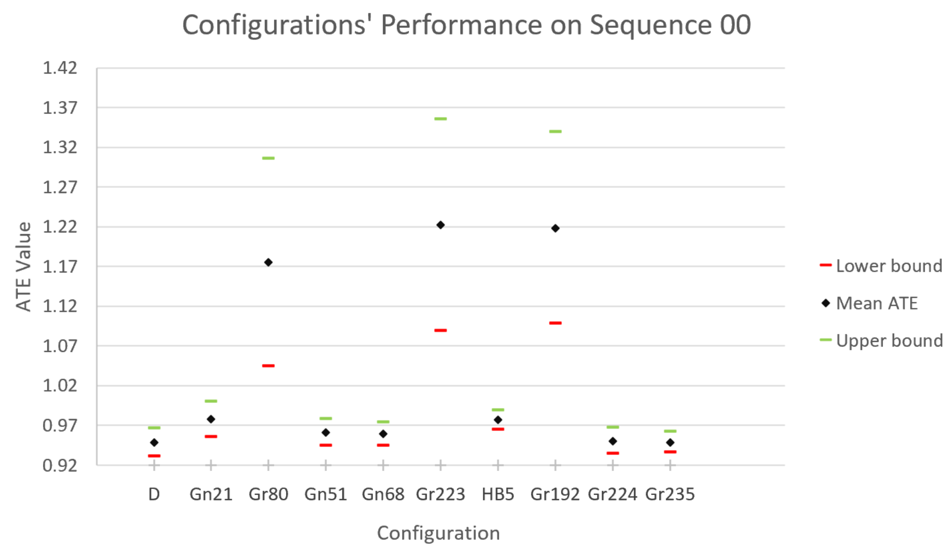
Table 4.5: Error reduction for each configuration candidate on each of the testing sequences.

Configuration	**Error reduction per sequence (%)**									
Name	S00	S01	S02	S03	S05	S06	S07	S08	S09	S10
Gen21	0.00	1.11	-15.75	-44.86	0.00	-27.01	0.16	12.78	28.11	-16.02
Grd80	-39.56	49.80	0.00	-23.97	0.00	-26.65	1.67	10.57	-169.48	0.00
Gen51	0.00	0.00	-16.21	-34.32	0.00	-21.72	0.37	13.49	28.65	-17.56
Gen68	0.00	0.00	0.00	-42.16	0.00	-26.51	0.00	14.73	60.16	-21.96
Grd223	-44.76	51.16	0.00	-23.63	0.00	-40.27	0.62	6.31	-162.47	0.00
HB5	0.00	0.00	-14.44	-48.36	0.00	-21.55	0.00	18.74	57.36	-21.36
Grd192	-43.05	49.42	5.14	-20.02	0.00	0.00	1.66	2.86	-181.29	0.00
Grd224	0.00	51.23	0.00	-18.65	0.00	0.00	3.21	0.25	-198.08	13.12
Grd235	0.00	50.84	0.00	-18.59	0.00	0.00	1.18	2.23	-185.36	2.08

A value of 0.00 means the result was found within the default execution's confidence interval.



(a)



(b)

Figure 4.1: The performance obtained from the configurations run on the unaltered KITTI sequences. The black dot represents the mean ATE value and the green and red lines, the upper and lower bounds, respectively. a) Sequence 03 shows a case where all the configurations had an increased ATE concerning the default execution (labeled as D). b) Sequence 00 is an example where most configurations showed no error reduction. The values obtained were within the confidence interval of the default execution (labeled as D)

Grd235) have an error reduction of over 50 % for sequence 01, there is a considerably significant ATE increment for sequence 09 (over 180 %). Additionally, the error reduction for sequences 07 and 08 is minimal[20].

[20] Less than 10 %

4.3.3 Final Results

Due to the results obtained in Section 4.3.1 and the capacity of the genetic algorithms to exploit the better-fitted results, we expected that a genetic approach would outperform other training algorithms during testing. However, the success rate (Table 4.4) demonstrates that a grid search provided the best solution given our constraints. We hypothesize that we need to increase the initial population of the genetic algorithm to obtain a higher success rate.

Some of the error reduction results (Table G.1) are so small that they could be considered proximal to the default execution's confidence interval. Additionally, the percentage of error increment for the cases where the configurations underperformed is higher than those where there was an error reduction. This increment means that the configurations candidates are ill-suited for general optimization.

Despite not finding a universal configuration that outperforms the default parameters, the results from testing indicate that when using grid search for a limited amount of time, there is an 80 % chance of either reducing the error or staying within a default configuration's confidence interval. Also, with the initialization parameters given, the genetic algorithm produced a 70 % chance of reducing the overall system's error. Nevertheless, there is also a considerably high chance that decreasing the error for a particular sequence might increase the ATE on another. Therefore, it might be better to use HPO to tune the SLAM's parameters for specific cases rather than a universal solution.

Chapter 5

Conclusions

In this work, we propose a methodology for implementing model-free, population-based hyperparameter optimization as a means to enhance SLAM and increase its performance. It has four components: environmental setup, parameter selection, benchmark selection, and algorithm selection. The environmental setup refers to the definition of constraints and modification of the SLAM system (in our case ORB-SLAM2). Tuners will be able to change all internal and hard-coded parameters through an external configuration file. The parameter selection requires a search space to be delimited. It also studies the influence of each parameter on the SLAM solution using Spearman's correlation coefficient to determine which parameters to select. For ORB-SLAM2, the ideal parameters to optimize are *ORBextractor.iniThFAST*, *ORBextractor.nLevels*, *GlobalBundleAdjustment.Iterations*, *ReconstructH.minParallax*, and *EraseObservation.minObservations*. Population-based algorithms were selected to optimize the SLAM system on an outdoor benchmark with available ground truth, such as the KITTI sequences. We designed the methodology so that anyone can export it to any other SLAM solution.

We adapted three existing performance evaluation metrics: optimality (Δo), underperformance (Δu), and success (Δs) rates to consider the information provided by the confidence intervals. We introduced a new performance metric, proximity rate (Δp). These metrics consider the information provided by the confidence intervals and offer qualitative data on the behavior of computed ATE compared to a default execution. The optimality rate determines the percentage for outperforming the default configuration. The underperformance rate is the percentage for exceeding the error computed for the default execution. The proximity rate signals the percentage for comparable performance concerning the ATE obtained from a default run. The information provided by these metrics is a percentile of the total tests. The success rate is the combined metric of optimality and proximity that evaluates the efficacy of a configuration. Together, these evaluation metrics provide information about the overall configuration performance on the SLAM solution.

ORB-SLAM2 was treated as a black-box function and trained with the selected algorithms in a modified sequence. Then, we evaluated the results on two different sets of test cases. We used the performance metrics proposed in the methodology to analyze the obtained configurations and compare them with the results from a default execution. We found that some of the population-based candidates tested on ORB-SLAM2 had a 70 % success rate in outperforming the default configuration, while the baseline algorithms showed an 80 % success rate. Despite not finding a set of hyperparameters that excels over the default parameters, applying HPO on SLAM is effective for case-specific executions.

We propose that our methodology for optimizing ORB-SLAM2 can effectively increase SLAM performance. We can see the performance increment throughout each tested configuration's success rates and error reduction (up to 60 % in a sequence). While parameter tuning is often a complicated and time-consuming process, this methodology optimizes a few parameters practically, removes the need for an expert tuner, and has a simple implementation. Theoretically, the methodology can be universal and is a viable approach for increasing the performance of any SLAM application.

5.1 Future Work

HPO is a rich study field that offers opportunities for future work. We have adapted an HPO approach for its use on ORB-SLAM2. It shows promise in enhancing SLAM performance via parameter optimization, but there are still many open questions to answer.

Parameter Space

Parameter space definition is critical for SLAM enhancement. The parameter space determines the limits of the parameter values. HPO algorithms need a defined search space to explore the possible parameter configurations efficiently. Section 3.2.2 defined a parameter search space that resulted in thousands of possible parameter combinations. Those combinations lead to computationally expensive algorithms that rely on factors to reduce runtime.

We plan to establish a stricter space definition metric to reduce the possible number of executions for the baseline algorithms. Increasing the quality of the search space also benefits the other training algorithms by augmenting the rate of finding local optima (i.e., a value that minimizes the error), which, in return, produces more precise results faster.

Parameter Relationships

In Section 3.2.3, we determined that the computational cost of optimization exponentially increases as the number of parameters increases. We used Spearman's correlation coefficient to find the parameters that had the most influence on the resulting ATE. Nevertheless, we assumed that all parameters were independent and presented no covariance with each other.

We also plan to add a metric to our methodology to study the covariance between parameters. This metric can change the parameter selection process as the influence of parameters affects the ATE produced and the performance of other parameters.

This way, the methodology introduced can become more accurate. We can hone the approach to produce more reliable results as we limit the number of parameters to optimize considering the covarying parameters.

Optimization Algorithms

Section 3.4, proposed using model-free algorithms to train our chosen SLAM solution. The methodology proposed resulted in the need for certain factors to aid the execution of some of the algorithms. For example, the parameter space made the grid search algorithm need a stopping criterion. We plan to produce enhancements to fine-tune the model-free algorithms for their application in SLAM. Some examples are tweaking the resource and budget parameters in HB, increasing the population size in the GA, or setting a better stopping criterion for random search.

Similarly, we plan to introduce model-based algorithms to our methodology. Using the information from previous executions to produce a surrogate model may help direct the search for more accurate configurations. That way, we could have a higher convergence rate into optimal solutions that enhance SLAM performance.

Other SLAM Solutions

We explored the application of the methodology on ORB-SLAM2. Theoretically, the optimization process relies on the training sequences and not the SLAM solution. Therefore, we assumed that the HPO approach for SLAM is viable and effective on other SLAM solutions. In the future, we plan to replicate and test this methodology to compare the effectiveness of HPO optimization in different SLAM approaches (e.g., DSO, LDSO) with the results obtained in this research.

Bibliography

[1] R. Elshawi, M. Maher, and S. Sakr, *Automated machine learning: State-of-the-art and open challenges*, 2019. arXiv: 1906.02287 [cs.LG].

[2] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: A versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[3] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[4] J. Ni, T. Gong, Y. Gu, J. Zhu, and X. Fan, "An improved deep residual network-based semantic simultaneous localization and mapping method for monocular vision robot," *Computational intelligence and neuroscience*, vol. 2020, 2020.

[5] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European conference on computer vision*, Springer, 2014, pp. 834–849.

[6] X. Gao, R. Wang, N. Demmel, and D. Cremers, "Ldso: Direct sparse odometry with loop closure," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 2198–2204.

[7] M. Labbé and F. Michaud, "Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[8] R. Storn, "On the usage of differential evolution for function optimization," in *Proceedings of North American Fuzzy Information Processing*, IEEE, 1996, pp. 519–523.

[9] L. Yang, F. Tan, A. Li, Z. Cui, Y. Furukawa, and P. Tan, "Polarimetric dense monocular slam," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3857–3866.

[10] P. L. N. Carrasco and G. O. Codina, "Visual odometry parameters optimization for autonomous underwater vehicles," *Instrumentation viewpoint*, vol. 15, 2013.

[11] T. Duckett *et al.*, "A genetic algorithm for simultaneous localization and mapping," 2003.

[12] A. Kostusiak and P. Skrzypczyński, "On the efficiency of population-based optimization in finding best parameters for rgb-d visual odometry," *Journal of Automation Mobile Robotics and Intelligent Systems*, vol. 13, 2019.

[13] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, 1309–1332, Dec. 2016, ISSN: 1941-0468. DOI: 10.1109/tro.2016.2624754. [Online]. Available: http://dx.doi.org/10.1109/TRO.2016.2624754.

[14] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[15] M. Andersson and M. Baerveldt, "Simultaneous localization and mapping for vehicles using orb-slam2," Master's thesis, 2018.

[16] S. K. Smit and A. E. Eiben, "Comparing parameter tuning methods for evolutionary algorithms," in *2009 IEEE congress on evolutionary computation*, IEEE, 2009, pp. 399–406.

[17] Z. Zheng, "Feature based monocular visual odometry for autonomous driving and hyperparameter tuning to improve trajectory estimation," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1453, 2020, p. 012 067.

[18] C. Zimmermann, T. Welschehold, C. Dornhege, W. Burgard, and T. Brox, "3d human pose estimation in rgbd images for robotic task learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 1986–1992.

[19] Z. Xian, X. He, J. Lian, X. Hu, and L. Zhang, "A bionic autonomous navigation system by using polarization navigation sensor and stereo camera," *Autonomous Robots*, vol. 41, no. 5, pp. 1107–1118, 2017.

[20] S. G. B. Rylander and B Gotshall, "Optimal population size and the genetic algorithm," *Population*, vol. 100, no. 400, p. 900, 2002.

[21] F. Berkenkamp, A. Krause, and A. P. Schoellig, "Bayesian optimization with safety constraints: Safe and automatic parameter tuning in robotics," *arXiv preprint arXiv:1602.04450*, 2016.

[22] I. A. Putra and P. Prajitno, "Parameter tuning of g-mapping slam (simultaneous localization and mapping) on mobile robot with laser-range finder 360° sensor," in *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, IEEE, 2019, pp. 148–153.

[23] A. Sehgal, A. Singandhupe, H. M. La, A. Tavakkoli, and S. J. Louis, "Lidar-monocular visual odometry with genetic algorithm for parameter optimization," in *International Symposium on Visual Computing*, Springer, 2019, pp. 358–370.

[24] T. N. Thanh, Y. Sakaguchi, H. Nagahara, and M. Yachida, "Stereo slam using two estimators," in *2006 IEEE International Conference on Robotics and Biomimetics*, 2006, pp. 19–24. DOI: 10.1109/ROBIO.2006.340253.

[25] A. Association for Advancing Automation, *What is visual slam technology and what is it used for?* May 2015. [Online]. Available: https://www.automate.org/blogs/what-is-visual-slam-technology-and-what-is-it-used-for.

[26] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Computer Vision, IEEE International Conference on*, IEEE Computer Society, vol. 3, 2003, pp. 1403–1403.

[27] I. Mahon, S. B. Williams, O. Pizarro, and M. Johnson-Roberson, "Efficient view-based slam using visual loop closures," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1002–1014, 2008.

[28] S. Kim and S.-Y. Oh, "Slam in indoor environments using omni-directional vertical and horizontal line features," *Journal of Intelligent and Robotic Systems*, vol. 51, no. 1, pp. 31–43, 2008.

[29] J. Shi *et al.*, "Good features to track," in *1994 Proceedings of IEEE conference on computer vision and pattern recognition*, IEEE, 1994, pp. 593–600.

[30] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2012.

[31] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, "An overview to visual odometry and visual slam: Applications to mobile robotics," *Intelligent Industrial Systems*, vol. 1, no. 4, pp. 289–311, 2015.

[32] C. A. C. Coello, *Learning and Intelligent Optimization*. Springer, 2011.

[33] M. Hosseinzadeh, K. Li, Y. Latif, and I. Reid, "Real-time monocular object-model aware sparse slam," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 7123–7129.

[34] S. Ishida, *How robots make maps- an intro to slam (simultaneous localisation and mapping)*, Aug. 2020. [Online]. Available: https://medium.com/swlh/how-robots-make-maps-an-intro-to-slam-simultaneous-localisation-and-mapping-37370c3e7dfe.

[35] K. Tateno, F. Tombari, I. Laina, and N. Navab, "Cnn-slam: Real-time dense monocular slam with learned depth prediction," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6243–6252.

[36] S. Ahn, J. Choi, N. L. Doh, and W. K. Chung, "A practical approach for ekf-slam in an indoor environment: Fusing ultrasonic sensors and stereo camera," *Autonomous robots*, vol. 24, no. 3, pp. 315–335, 2008.

[37] L. Mu, P. Yao, Y. Zheng, K. Chen, F. Wang, and N. Qi, "Research on slam algorithm of mobile robot based on the fusion of 2d lidar and depth camera," *IEEE Access*, vol. 8, pp. 157 628–157 642, 2020.

[38] S. García, M. E. López, R. Barea, L. M. Bergasa, A. Gómez, and E. J. Molinos, "Indoor slam for micro aerial vehicles control using monocular camera and sensor fusion," in *2016 international conference on autonomous robot systems and competitions (ICARSC)*, IEEE, 2016, pp. 205–210.

[39] S. Jo, H. Choi, and E. Kim, "Ceiling vision based slam approach using sensor fusion of sonar sensor and monocular camera," in *2012 12th International Conference on Control, Automation and Systems*, IEEE, 2012, pp. 1461–1464.

[40] S. Das, "Simultaneous localization and mapping (slam) using rtab-map," *arXiv preprint arXiv:1809.02989*, 2018.

[41] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*, Ieee, 2011, pp. 2564–2571.

[42] G. Grisetti, R. Kümmerle, H. Strasdat, and K. Konolige, "G2o: A general framework for (hyper) graph optimization," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China*, 2011, pp. 9–13.

[43] N. Ragot, R. Khemmar, A. Pokala, R. Rossi, and J. Ertaud, "Benchmark of visual slam algorithms: Orb-slam2 vs rtab-map*," in *2019 Eighth International Conference on Emerging Security Technologies (EST)*, 2019, pp. 1–6.

[44] A. M. Webb, G. Brown, and M. Luján, "Orb-slam-cnn: Lessons in adding semantic map construction to feature-based slam," in *Annual Conference Towards Autonomous Robotic Systems*, Springer, 2019, pp. 221–235.

[45] S. Scheideman, N. Ray, and H. Zhang, "A flexible method for performance evaluation of robot localization," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 8302–8308.

[46] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 573–580.

[47] O. Wulf, A. Nuchter, J. Hertzberg, and B. Wagner, "Ground truth evaluation of large urban 6d slam," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2007, pp. 650–657.

[48] Z. Chen, "Visual-inertial slam extrinsic parameter calibration based on bayesian optimization," 2018.

[49] J. Sturm, *Rgb-d slam dataset and benchmark*, 2012.

[50] R. Andonie, "Hyperparameter optimization in learning systems," *Journal of Membrane Computing*, vol. 1, no. 4, pp. 279–291, 2019.

[51] P. P. Ippolito, *Hyperparameters optimization*, Sep. 2019. [Online]. Available: fromhttps://towardsdatascience.com/hyperparameters-optimization-526348bb8e2d.

[52] R. Dwivedi, *Introduction to model hyperparameter and tuning in machine learning*, May 2020. [Online]. Available: https://www.analyticssteps.com/blogs/introduction-model-hyperparameter-and-tuning-machine-learning.

[53] A. Bissuel, *Hyperparameter optimization algorithms: A short review*, Apr. 2019. [Online]. Available: https://medium.com/criteo-engineering/hyper-parameter-optimization-algorithms-2fe447525903.

[54] C. Huang, Y. Li, and X. Yao, "A survey of automatic parameter tuning methods for metaheuristics," *IEEE transactions on evolutionary computation*, vol. 24, no. 2, pp. 201–216, 2019.

[55] W. Koehrsen, *Hyperparameter tuning the random forest in python*, Jan. 2018. [Online]. Available: https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74.

[56] L. Li, *Why does no one use advanced hyperparameter tuning?* Oct. 2020. [Online]. Available: https://www.determined.ai/blog/why-does-no-one-use-advanced-hp-tuning.

[57] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, "Hyperparameter optimization for machine learning models based on bayesian optimization," *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019.

[58] F. Hutter, "Automated configuration of algorithms for solving hard computational problems," PhD thesis, University of British Columbia, 2009.

[59] Y. Kim, H. Son, Y. J. Lee, and S. Sung, "Integrated navigation algorithm using velocity incremental vector approach with orb-slam and inertial measurement," *The Transactions of the Korean Institute of Electrical Engineers*, vol. 68, no. 1, pp. 189–198, 2019.

[60] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *International conference on learning and intelligent optimization*, Springer, 2011, pp. 507–523.

[61] X. Xiao, B. Liu, G. Warnell, J. Fink, and P. Stone, "Appld: Adaptive planner parameter learning from demonstration," *arXiv preprint arXiv:2004.00116*, 2020.

[62] M. Annunziato and S. Pizzuti, "Adaptive parameterization of evolutionary algorithms driven by reproduction and competition," in *Proceedings of the European Symposium on Intelligent Techniques (ESIT 2000), Aachen, Germany*, 2000, pp. 31–35.

[63] J. Cheng, G. G. Yen, and G. Zhang, "A many-objective evolutionary algorithm with enhanced mating and environmental selections," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 4, pp. 592–605, 2015.

[64] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, Ieee, 1995, pp. 39–43.

[65] R. Rutenbar, "Simulated annealing algorithms: An overview," *IEEE Circuits and Devices Magazine*, vol. 5, no. 1, pp. 19–26, 1989. DOI: 10.1109/101.17235.

[66] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[67] M. Birattari and J. Kacprzyk, *Tuning metaheuristics: a machine learning perspective*. Springer, 2009, vol. 197.

[68] E.-G. Talbi, *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009, vol. 74.

[69] M. Birattari, T. Stützle, L. Paquete, K. Varrentrapp, *et al.*, "A racing algorithm for configuring metaheuristics.," in *Gecco*, vol. 2, 2002.

[70] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in neural information processing systems*, 2011, pp. 2546–2554.

[71] P. T. Sivaprasad, F. Mai, T. Vogels, M. Jaggi, and F. Fleuret, "Optimizer benchmarking needs to account for hyperparameter tuning," in *International Conference on Machine Learning*, PMLR, 2020, pp. 9036–9045.

[72] P. Y. Kuo, H. Du, L. A. Corkan, K. Yang, and J. S. Lindsey, "A planning module for performing grid search, factorial design, and related combinatorial studies on an automated chemistry workstation," *Chemometrics and intelligent laboratory systems*, vol. 48, no. 2, pp. 219–234, 1999.

[73] I. Syarif, A. Prugel-Bennett, and G. Wills, "Svm parameter optimization using grid search and genetic algorithm to improve classification performance," *Telkomnika*, vol. 14, no. 4, p. 1502, 2016.

[74] D. S. Soper, "Greed is good: Rapid hyperparameter optimization and model selection using greedy k-fold cross validation," *Electronics*, vol. 10, no. 16, p. 1973, 2021.

[75] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, 2020.

[76] C. Sun, D. Liu, and C. Yang, "Model-free unsupervised learning for optimization problems with constraints," in *2019 25th Asia-Pacific Conference on Communications (APCC)*, IEEE, 2019, pp. 392–397.

[77] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 281–305, 2012.

[78] Z. B. Zabinsky, "Random search algorithms," *Wiley encyclopedia of operations research and management science*, 2010.

[79] H. Mania, A. Guy, and B. Recht, *Simple random search provides a competitive approach to reinforcement learning*, 2018. arXiv: 1803.07055 [cs.LG].

[80] C. B. Yazici, *Continuous genetic algorithm from scratch with python*, Oct. 2019. [Online]. Available: https://towardsdatascience.com/continuous-genetic-algorithm-from-scratch-with-python-ff29deedd099.

[81] B. Shekar and G. Dagnew, "Grid search-based hyperparameter tuning and classification of microarray cancer data," in *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)*, IEEE, 2019, pp. 1–8.

[82] R. G. Mantovani, A. L. D. Rossi, J. Vanschoren, B. Bischl, and A. C. P. L. F. de Carvalho, "Effectiveness of random search in svm hyper-parameter tuning," in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–8. DOI: 10.1109/IJCNN.2015.7280664.

[83] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[84] W. Ponweiser, T. Wagner, and M. Vincze, "Clustered multiple generalized expected improvement: A novel infill sampling criterion for surrogate models," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, IEEE, 2008, pp. 3515–3522.

[85] D. Huang, T. T. Allen, W. I. Notz, and N. Zeng, "Global optimization of stochastic black-box systems via sequential kriging meta-models," *Journal of global optimization*, vol. 34, no. 3, pp. 441–466, 2006.

[86] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Automatic hyperparameter tuning for black-box lidar odometry," in *IEEE International Conference on Robotics and Automation. IEEE*, 2021.

[87] F. Hutter, H. H. Hoos, K. Leyton-Brown, and K. P. Murphy, "An experimental investigation of model-based parameter optimisation: Spo and beyond," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, 2009, pp. 271–278.

[88] E. Brochu, V. M. Cora, and N. De Freitas, "A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv preprint arXiv:1012.2599*, 2010.

[89] M. D. Hoffman, E. Brochu, and N. de Freitas, "Portfolio allocation for bayesian optimization.," in *UAI*, Citeseer, 2011, pp. 327–336.

[90] J. Mockus, V. Tiesis, and A. Zilinskas, "The application of bayesian methods for seeking the extremum," *Towards global optimization*, vol. 2, no. 117-129, p. 2, 1978.

[91] M. Bahmani, *Hyperband and bohb: Understanding state of the art hyperparameter optimization algorithms*, May 2021. [Online]. Available: https://neptune.ai/blog/hyperband-and-bohb-understanding-state-of-the-art-hyperparameter-optimization-algorithms.

[92] S. Falkner, A. Klein, and F. Hutter, "Bohb: Robust and efficient hyperparameter optimization at scale," *arXiv preprint arXiv:1807.01774*, 2018.

[93] C. Audet and W. Hare, "Derivative-free and blackbox optimization," 2017.

[94] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, "Adaptive hyperparameter tuning for black-box lidar odometry," *arXiv preprint arXiv:2107.00275*, 2021.

[95] H. Juan-Rou and W. Zhan-Qing, "The implementation of imu/stereo vision slam system for mobile robot," in *2020 27th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS)*, IEEE, 2020, pp. 1–4.

[96] D. Belter, M. Nowicki, and P. Skrzypczyński, "Modeling spatial uncertainty of point features in feature-based rgb-d slam," *Machine Vision and Applications*, vol. 29, no. 5, pp. 827–844, 2018.

[97] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6765–6816, 2017.

[98] G. Lan, J. M. Tomczak, D. M. Roijers, and A. Eiben, "Time efficiency in optimization with a bayesian-evolutionary algorithm," *arXiv preprint arXiv:2005.04166*, 2020.

[99] J. Carnahan and R. Sinha, "Nature's algorithms [genetic algorithms]," *IEEE Potentials*, vol. 20, no. 2, pp. 21–24, 2001. DOI: 10.1109/45.954644.

[100] F. J. Rodriguez, C. Garcia-Martinez, and M. Lozano, "Hybrid metaheuristics based on evolutionary algorithms and simulated annealing: Taxonomy, comparison, and synergy test," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 6, pp. 787–800, 2012. DOI: 10.1109/TEVC.2012.2182773.

[101] K. Jamieson and A. Talwalkar, "Non-stochastic best arm identification and hyperparameter optimization," in *Artificial Intelligence and Statistics*, 2016, pp. 240–248.

[102] M. Binder, J. Moosbauer, J. Thomas, and B. Bischl, "Multi-objective hyperparameter tuning and feature selection using filter ensembles," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, 2020, pp. 471–479.

[103] D. Vrajitoru, "Large population or many generations for genetic algorithms? implications in information retrieval," in *Soft Computing in Information Retrieval*, Springer, 2000, pp. 199–222.

[104] Y. Wang, Z. Cai, Y. Zhou, and Z. Fan, "Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique," *Structural and Multidisciplinary Optimization*, vol. 37, no. 4, pp. 395–413, 2009.

[105] N Botteghi, B. Sirmacek, R Schulte, M Poel, and C Brune, "Reinforcement learning helps slam: Learning to build maps," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 43, pp. 329–335, 2020.

[106] R. S. Sutton, "A special issue of machine learning on reinforcement learning," *Machine learning*, vol. 8, 1992.

[107] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[108] A. Bar-Hillel, A. Di-Nur, L. Ein-Dor, R. Gilad-Bachrach, and Y. Ittach, "Workstation capacity tuning using reinforcement learning," in *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*, 2007, pp. 1–11.

[109] A. Chaurasia, *Comparing modern scalable hyperparameter tuning methods*, Sep. 2020. [Online]. Available: https://towardsdatascience.com/comparing-modern-scalable-hyperparameter-tuning-methods-dfe9661e947f.

[110] S. Cheng, B. Liu, T. Ting, Q. Qin, Y. Shi, and K. Huang, "Survey on data science with population-based algorithms," *Big Data Analytics*, vol. 1, no. 1, pp. 1–20, 2016.

[111] K. De Jong, "Learning with genetic algorithms: An overview," *Machine learning*, vol. 3, no. 2-3, pp. 121–138, 1988.

[112] F. Streichert, "Introduction to evolutionary algorithms," *paper to be presented Apr*, vol. 4, 2002.

[113] I. BoussaïD, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information sciences*, vol. 237, pp. 82–117, 2013.

[114] A. N. Sloss and S. Gustafson, "2019 evolutionary algorithms review," *Genetic Programming Theory and Practice XVII*, pp. 307–344, 2020.

[115] T. Sølund, "Towards plug-n-play robot guidance: Advanced 3d estimation and pose estimation in robotic applications," 2017.

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

[117] F. Glover and M. Laguna, "Tabu search," in *Handbook of combinatorial optimization*, Springer, 1998, pp. 2093–2229.

[118] D. Whitley, "A genetic algorithm tutorial," *Statistics and computing*, vol. 4, no. 2, pp. 65–85, 1994.

[119] B. Xi, Z. Liu, M. Raghavachari, C. H. Xia, and L. Zhang, "A smart hill-climbing algorithm for application server configuration," in *Proceedings of the 13th international conference on World Wide Web*, 2004, pp. 287–296.

[120] Y. Hajizadeh, V. Demyanov, L. Mohamed, and M. Christie, "Comparison of evolutionary and swarm intelligence methods for history matching and uncertainty quantification in petroleum reservoir models," in *Intelligent Computational Optimization in Engineering*, Springer, 2011, pp. 209–240.

[121] N. Nagarajan, "Multi-objective optimisation of rtab-map parameters using genetic algorithm for indoor 2d slam," PhD thesis, 2020.

[122] D. E. Goldberg and J. H. Holland, “Genetic algorithms and machine learning,” 1988.

[123] T. V. Mathew, “Genetic algorithm,” *Report submitted at IIT Bombay*, 2012.

[124] S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton, “Optimizing deep learning hyper-parameters through an evolutionary algorithm,” in *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*, 2015, pp. 1–5.

[125] O. Ghasemalizadeh, M. Khaleghian, and S. Taheri, “A review of optimization techniques in artificial networks,” *International Journal of Advanced Research*, vol. 4, pp. 1668–1686, Sep. 2016. DOI: 10.21474/IJAR01/1627.

[126] K. Jamieson and B. Recht, *Embracing the random*, Jun. 2016. [Online]. Available: https://www.argmin.net/2016/06/23/hyperband/.

[127] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, M. Hardt, B. Recht, and A. Talwalkar, “A system for massively parallel hyperparameter tuning,” *arXiv preprint arXiv:1810.05934*, 2018.

[128] L. H. Franc, *The hyperband algorithm*, May 2018. [Online]. Available: https://louishenrifranc.github.io/mathblog/2017/05/08/hyperband/.

[129] A. Gad, *Genetic algorithm implementation in python*, Jul. 2018. [Online]. Available: https://towardsdatascience.com/genetic-algorithm-implementation-in-python-5ab67bb124a6.

[130] A. Abraham, *A (slightly) better budget allocation for hyperband*, Apr. 2020. [Online]. Available: https://medium.com/data-from-the-trenches/a-slightly-better-budget-allocation-for-hyperband-bbd45af14481.

[131] S. Scheideman, “Estimating robot localization error using visual marker pose estimation,” 2019.

[132] S. Y. Loo, A. J. Amiri, S. Mashohor, S. H. Tang, and H. Zhang, “Cnn-svo: Improving the mapping in semi-direct visual odometry using single-image depth prediction,” in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 5218–5223.

[133] D. P. Bovet, P. Crescenzi, and D Bovet, *Introduction to the Theory of Complexity*. Prentice Hall London, 1994.

[134] P. Liashchynskyi and P. Liashchynskyi, “Grid search, random search, genetic algorithm: A big comparison for nas,” *arXiv preprint arXiv:1912.06059*, 2019.

[135] J. Freyberger and M. A. Masten, “A practical guide to compact infinite dimensional parameter spaces,” *Econometric Reviews*, vol. 38, no. 9, pp. 979–1006, 2019.

[136] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[137] A. Lehman, N. O’Rourke, L. Hatcher, and E. Stepanski, *JMP for basic univariate and multivariate statistics: methods for researchers and social scientists*. Sas Institute, 2013.

[138] S. Cheusheva, *How to do spearman correlation in excel*, Mar. 2021. [Online]. Available: https://www.ablebits.com/office-addins-blog/2019/01/30/spearman-rank-correlation-excel/.

[139] C. H. Olsen, J. T. Ottesen, R. C. Smith, and M. S. Olufsen, "Parameter subset selection techniques for problems in mathematical biology," *Biological cybernetics*, vol. 113, no. 1, pp. 121–138, 2019.

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

[141] A. Hayes, G. Scott, Ed., Oct. 2021. [Online]. Available: https://www.investopedia.com/terms/c/confidenceinterval.asp.

[142] S. F. O'Brien and Q. L. Yi, "How do i interpret a confidence interval?" *Transfusion*, vol. 56, no. 7, pp. 1680–1683, 2016.

[143] J. Di Stefano, "A confidence interval approach to data analysis," *Forest Ecology and Management*, vol. 187, no. 2-3, pp. 173–183, 2004.

[144] N. Pandis, "Statistical inference with confidence intervals," *American journal of orthodontics and dentofacial orthopedics*, vol. 147, no. 5, pp. 632–634, 2015.

[145] J. Sim and N. Reid, "Statistical inference by confidence intervals: Issues of interpretation and utilization," *Physical Therapy*, vol. 79, no. 2, pp. 186–195, 1999.

[146] D. Selvi, D. Piga, and A. Bemporad, "Towards direct data-driven model-free design of optimal controllers," in *2018 European Control Conference (ECC)*, IEEE, 2018, pp. 2836–2841.

[147] A. Morris and F. Cushman, "Model-free rl or action sequences?" *Frontiers in Psychology*, vol. 10, p. 2892, 2019, ISSN: 1664-1078. DOI: 10.3389/fpsyg.2019.02892. [Online]. Available: https://www.frontiersin.org/article/10.3389/fpsyg.2019.02892.

[148] Z. Hou and S. Jin, *Model free adaptive control: theory and applications*. CRC press, 2019.

[149] K. J. Miller, A. Shenhav, and E. A. Ludvig, "Habits without values.," *Psychological review*, vol. 126, no. 2, p. 292, 2019.

[150] N. Gireesh, *Model-based offline reinforcement learning (morel)*, Jun. 2020. [Online]. Available: https://medium.com/analytics-vidhya/model-based-offline-reinforcement-learning-morel-f5cd991d9fd5.

[151] K. Asadi, "Strengths, weaknesses, and combinations of model-based and model-free reinforcement learning," *Department of Computing Science University of Alberta*, 2015.

[152] "Baseline algorithm," in *Encyclopedia of Biometrics*, S. Z. Li and A. Jain, Eds. Boston, MA: Springer US, 2009, pp. 60–60, ISBN: 978-0-387-73003-5. DOI: 10.1007/978-0-387-73003-5_538. [Online]. Available: https://doi.org/10.1007/978-0-387-73003-5_538.

[153] M. Ohsaki and M. Yamakawa, "Stopping rule of multi-start local search for structural optimization," *Structural and Multidisciplinary Optimization*, vol. 57, no. 2, pp. 595–603, 2018.

[154] K. Vo, T. Pham, D. N. Nguyen, H. H. Kha, and E. Dutkiewicz, "Subject-independent erp-based brain–computer interfaces," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 26, no. 4, pp. 719–728, 2018.

[155] A. Demircioglu, D. Horn, T. Glasmachers, B. Bischl, and C. Weihs, "Fast model selection by limiting svm training times," *arXiv preprint arXiv:1602.03368*, 2016.

[156] M. Appel, R Labarre, and D Radulovic, "On accelerated random search," *SIAM Journal on Optimization*, vol. 14, no. 3, pp. 708–731, 2004.

[157] N. Maheswaranathan, L. Metz, G. Tucker, D. Choi, and J. Sohl-Dickstein, "Guided evolutionary strategies: Augmenting random search with surrogate gradients," in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, Jun. 2019, pp. 4264–4273. [Online]. Available: https://proceedings.mlr.press/v97/maheswaranathan19a.html.

[158] A. C. Florea and R. Andonie, "A dynamic early stopping criterion for random search in svm hyperparameter optimization," in *Artificial Intelligence Applications and Innovations*, L. Iliadis, I. Maglogiannis, and V. Plagianakos, Eds., Cham: Springer International Publishing, 2018, pp. 168–180, ISBN: 978-3-319-92007-8.

[159] M. Ohsaki, "Random search method based on exact reanalysis for topology optimization of trusses with discrete cross-sectional areas," *Computers & Structures*, vol. 79, no. 6, pp. 673–679, 2001.

[160] M. Ohsaki and M. Katsura, "A random sampling approach to worst-case design of structures," *Structural and Multidisciplinary Optimization*, vol. 46, no. 1, pp. 27–39, 2012.

[161] J. Feng and W. Z. Shen, "Solving the wind farm layout optimization problem using random search algorithm," *Renewable Energy*, vol. 78, pp. 182–192, 2015, ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2015.01.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960148115000129.

[162] D. Simon, *Evolutionary optimization algorithms*. John Wiley & Sons, 2013.

[163] T. Weise, Y. Wu, R. Chiong, K. Tang, and J. Lässig, "Global versus local search: The impact of population sizes on evolutionary algorithm performance," *Journal of Global Optimization*, vol. 66, no. 3, pp. 511–534, 2016.

[164] G. Minetti and H. Alfonso, "Variable size population in parallel evolutionary algorithms," in *5th International Conference on Intelligent Systems Design and Applications (ISDA'05)*, 2005, pp. 350–355. DOI: 10.1109/ISDA.2005.99.

[165] A. Simões and E. Costa, "On biologically inspired genetic operators: Transformation in the standard genetic algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2001, pp. 584–591.

[166] G. Alvarez, "Can we make genetic algorithms work in high-dimensionality problems," *SEP–112*, pp. 195–212, 2002.

[167] C. Mihail-Bogdan, I. Constantin, and N. Horia, "The influence of genetic algorithm parameters over the efficiency of the energy consumption estimation in a low–energy building," *Energy Procedia*, vol. 85, pp. 99–108, 2016.

[168] G. Mosetti, C. Poloni, and B. Diviacco, "Optimization of wind turbine positioning in large windfarms by means of a genetic algorithm," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 51, no. 1, pp. 105–116, 1994.

[169] D. Akdemir, J. I. Sanchez, and J.-L. Jannink, "Optimization of genomic selection training populations with a genetic algorithm," *Genetics Selection Evolution*, vol. 47, no. 1, pp. 1–10, 2015.

[170] M. Bhargava, R. Mehta, C. D. Adhikari, and K Sivanathan, "Towards development of performance metrics for benchmarking slam algorithms," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1964, 2021, p. 062 115.

[171] M. Wistuba, N. Schilling, and L. Schmidt-Thieme, "Hyperparameter search space pruning–a new component for sequential model-based hyperparameter optimization," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2015, pp. 104–119.

[172] J. Klippenstein and H. Zhang, "Performance evaluation of visual slam using several feature extractors," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2009, pp. 1574–1581.

[173] G. Hu, K. Khosoussi, and S. Huang, "Towards a reliable slam back-end," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2013, pp. 37–43.

[174] G. Jiang, L. Yin, S. Jin, C. Tian, X. Ma, and Y. Ou, "A simultaneous localization and mapping (slam) framework for 2.5 d map building based on low-cost lidar and vision fusion," *Applied Sciences*, vol. 9, no. 10, p. 2105, 2019.

[175] M. F. Fallon, H. Johannsson, M. Kaess, J. Folkesson, H. McClelland, B. J. Englot, F. S. Hover, and J. J. Leonard, "Simultaneous localization and mapping in marine environments," in *Marine Robot Autonomy*, Springer, 2013, pp. 329–372.

[176] D. H. Jones, *Statistical methods*, 1994.

[177] G. Polya, *How to solve it: A new aspect of mathematical method*. Princeton university press, 2004, vol. 85.

[178] G. Polya, *Mathematical Discovery, 1962*. John Wiley & Sons, 1962.

[179] S. H. Zanakis and J. R. Evans, "Heuristic "optimization": Why, when, and how to use it," *Interfaces*, vol. 11, no. 5, pp. 84–91, 1981.

[180] T. Domhan, J. T. Springenberg, and F. Hutter, "Speeding up automatic hyper-parameter optimization of deep neural networks by extrapolation of learning curves," in *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[181] F. Ramsey and D. Schafer, *The statistical sleuth: a course in methods of data analysis*. Cengage Learning, 2012.

Appendix A: ORB-SLAM2 Parameters and Values

Name	Value
Tracking	
ORBextractor.nFeatures	2000
ORBextractor.scaleFactor	1.2
ORBextractor.nLevels	8
ORBextractor.iniThFAST	20
ORBextractor.minThFAST	7
Tracking.MonocularInitialization.minTrackedPoints	100
Tracking.MonocularInitialization.Initializer.mMaxIterations	200
Tracking.MonocularInitialization.Initializer.mSigma	1
Tracking.MonocularInitialization.OrbMatcher.LowesRatio	0.9
Tracking.MonocularInitialization.OrbMatcher.SearchForInitialization.windowSize	100
Tracking.MonocularInitialization.minMatchesBetweenFrames	100
Tracking.MonocularInitialization.Initializer.FindHomography.th	5.991
Tracking.MonocularInitialization.Initializer.FindFundamental.th	3.841
Tracking.MonocularInitialization.Initializer.FindFundamental.thScore	5.991
Tracking.MonocularInitialization.Initializer.Initialize.RH	0.4
Tracking.MonocularInitialization.Initializer.ReconstructH.minTriagulated	50
Tracking.MonocularInitialization.Initializer.ReconstructH.minParallax	1
Tracking.MonocularInitialization.Initializer.ReconstructH.checkRT.th	4
Tracking.MonocularInitialization.Initializer.ReconstructH.countInliersTh1	0.75
Tracking.MonocularInitialization.Initializer.ReconstructH.countInliersTh2	0.9
Tracking.MonocularInitialization.Initializer.ReconstructF.minTriagulated	50
Tracking.MonocularInitialization.Initializer.ReconstructF.minParallax	1
Tracking.MonocularInitialization.Initializer.ReconstructF.checkRT.th	4
Tracking.MonocularInitialization.Initializer.ReconstructF.countInliersTh1	0.7
Tracking.MonocularInitialization.Initializer.ReconstructF.countInliersTh2	0.9
Tracking.MonocularInitialization.Initializer.ReconstructF.nSimilarTh	1
Tracking.CreateInitialMapMonocular.GlobalBundleAdjustment.Iterations	20
Tracking.CreateInitialMapMonocular.MinTrackedMapPoints	100
Tracking.CreateInitialMapMonocular.TrackedMapPoints.minObs	1
Tracking.Track.TrackReferenceKeyFrame.OrbMatcher.LowesRatio	0.7
Tracking.Track.TrackReferenceKeyFrame.nnmatches	15
Tracking.Track.TrackReferenceKeyFrame.PoseOptimization.minInitialCorrespondences	3
Tracking.Track.TrackReferenceKeyFrame.PoseOptimization.robustHuberKernalDelta	5.991
Tracking.Track.TrackReferenceKeyFrame.PoseOptimization.minNumberOfEdges	10
Tracking.Track.TrackReferenceKeyFrame.optminMatches	10
Tracking.Track.TrackWithMotionModel.OrbMatcher.LowesRatio	0.9
Tracking.Track.TrackWithMotionModel.SearchByProjection.th	7
Tracking.Track.TrackWithMotionModel.MinMatches	20
Tracking.Track.TrackWithMotionModel.SearchByProjection.th2	15
Tracking.Track.TrackWithMotionModel.MinMapMatches	10
Tracking.Track.Relocalization.KeyframeDatabase.DetectRelocalizationCandidates.minCommonN	0.8
Tracking.Track.Relocalization.OrbMatcher1.LowesRatio	0.75
Tracking.Track.Relocalization.MinMatches	15
Tracking.Track.Relocalization.PnPSolver.SetRansacParameters.probablity	0.99

Tracking.Track.Relocalization.PnPSolver.SetRansacParameters.minInliers	10
Tracking.Track.Relocalization.PnPSolver.SetRansacParameters.maxIterations	300
Tracking.Track.Relocalization.PnPSolver.SetRansacParameters.minSet	4
Tracking.Track.Relocalization.PnPSolver.SetRansacParameters.epsilon	0.5
Tracking.Track.Relocalization.PnPSolver.SetRansacParameters.th2	5.991
Tracking.Track.Relocalization.OrbMatcher2.LowesRatio	0.9
Tracking.Track.Relocalization.ransacIterations	5
Tracking.Track.Relocalization.initialMinInliers	10
Tracking.Track.Relocalization.secondMinInliers	50
Tracking.Track.Relocalization.SearchByProjection1.th	10
Tracking.Track.Relocalization.SearchByProjection1.orbdist	100
Tracking.Track.Relocalization.SearchByProjection2.th	3
Tracking.Track.Relocalization.SearchByProjection2.orbdist	64
Tracking.Track.UpdateLocalMap.UpdateLocalKeyFrames.maxLocal	80
Tracking.Track.UpdateLocalMap.UpdateLocalKeyFrames.covisibility	10
Tracking.Track.TrackLocalMap.inlierThres	30
Tracking.Track.TrackLocalMap.inlierThresReloc	50
Tracking.Track.voMatchThres	1
Tracking.Track.NeedNewFrame.nMaxFrames	20
Tracking.Track.NeedNewFrame.referenceKeyFrameNminObs	3
Tracking.Track.NeedNewFrame.thRefRatio	0.9
Tracking.Track.NeedNewFrame.thCurrentFameTracks	15
Tracking.Track.thLostToFast	5

LoopClosing

LoopClosing.DetectLoops.KeyframesPast	10
LoopClosing.KeyFrameDatabase.minCommonWordsRatio	0.8
LoopClosing.KeyFrameDatabase.minScoreToRetainRatio	0.75
LoopClosing.KeyFrameDatabase.numBestCovisbility	10
LoopClosing.ComputeSim3.OrbMatcher.LowesRatio	0.75
LoopClosing.ComputeSim3.OrbMatcher.SearchByBoWMinMatches	20
LoopClosing.ComputeSim3.OrbMatcher.SearchBySim3.threshold	7.5
LoopClosing.ComputeSim3.OrbMatcher.SearchByProjection.threshold	10
LoopClosing.ComputeSim3.Sim3Solver.RansacProbability	0.99
LoopClosing.ComputeSim3.Sim3Solver.RansacMinInliers	20
LoopClosing.ComputeSim3.Sim3Solver.RansacMaxIterations	300
LoopClosing.ComputeSim3.RansacIteration	5
Optimizer.OptimizeSim3.th2	10
Optimizer.OptimizeSim3.iterations	5
Optimizer.OptimizeSim3.robustHuberKernelDelta	sqrt(th2)
LoopClosing.ComputeSim3.OptimizeSim3NumberInliers	20
LoopClosing.ComputeSim3.nTotalMatchTh	40
LoopClosing.CorrectLoop.SearchAndFuse.OrbMatcher.LowesRatio	0.8
LoopClosing.CorrectLoop.SearchAndFuse.OrbMatcher.Fuse.threshod	4
Optimizer.OptimizeEssentialGraph.minFeat	100
Optimizer.OptimizeEssentialGraph.iterations	20

Optimizer.GlobalBundleAdjustment.iterations	10
Optimizer.GlobalBundleAdjustment.robustHuberKernelDelta	2.449286

LocalMapping

LocalMapping.MapPointCulling.foundPercentCorrect	25
LocalMapping.MapPointCulling.cnThObs	3
LocalMapping.MapPointCulling.safeFromCullingCount	2
LocalMapping.MapPointCulling.recentlyAddedCount	3
LocalMapping.MapPointCulling.cullInGeneral	2
LocalMapping.CreateNewMapPoints.OrbMatcher	0.6
LocalMapping.CreateNewMapPoints.OrbMatcher.SearchForTriangulation.ThMultiplier	100
LocalMapping.CreateNewMapPoints.OrbMatcher.CheckDistEpipolarLine	3.84
LocalMapping.CreateNewMapPoints.nn	20
LocalMapping.CreateNewMapPoints.ratioFactorMultiplier	1.5
LocalMapping.CreateNewMapPoints.sigmaSquared2Multiplier	5.991
LocalMapping.SearchInNeighbours.nn	20
LocalMapping.SearchInNeighbours.nnSecondNeighbours	5
LocalMapping.SearchInNeighbours.OrbMatcher	0.6
LocalMapping.LocalBundleAdjustment.iterationsWithOutliers	5
LocalMapping.LocalBundleAdjustment.iterationsWithoutOutliers	10
LocalMapping.LocalBundleAdjustment.numberFramesTh	3
LocalMapping.LocalBundleAdjustment.thHuberMono	5.991
LocalMapping.LocalBundleAdjustment.chi2ErrorTh	5.991
LocalMapping.KeyFrameCulling.nObsTh	3
LocalMapping.KeyFrameCulling.redundantTh	0.9

Misc

Keyframe.UpdateConnections.threshold	15
Keyframe.ComputeBow	4
OrbMatcher.TH_HIGH	100
OrbMatcher.TH_LOW	50
OrbMatcher.HISTO_LENGTH	30
MapPoint.EraseObservation.minObservations	3

Appendix B: Spearman Correlation Full Results

Parameter	Spearman Value
ORBextractor.iniThFAST	0.753246753
ORBextractor.nLevels	0.649350649
Tracking.CreateInitialMapMonocular.GlobalBundleAdjustment.Iterations	0.558441558
Tracking.MonocularInitialization.Initializer.ReconstructH.minParallax	0.519480519
OrbMatcher.HISTO_LENGTH	0.506493506
MapPoint.EraseObservation.minObservations	0.469622332
LocalMapping.SearchInNeighbours.nnSecondNeighbours	0.467532468
Tracking.Track.Relocalization.initialMinInliers	0.461038961
LocalMapping.CreateNewMapPoints.OrbMatcher.SearchForTriangulation.ThMultiplier	0.444155844
Tracking.Track.Relocalization.PnPSolver.SetRansacParameters.epsilon	0.409090909
OrbMatcher.TH_LOW	0.341558442
Tracking.Track.voMatchThres	0.335064935
Tracking.Track.TrackWithMotionModel.SearchByProjection.th	0.331451158
Tracking.Track.Relocalization.KeyframeDatabase.DetectRelocalizationCandidates.minCommonMultiplier	0.324675325
LocalMapping.CreateNewMapPoints.nn	0.323376623
Tracking.Track.Relocalization.OrbMatcher1.LowesRatio	0.292207792
Tracking.MonocularInitialization.Initializer.FindFundamental.th	0.288311688
LocalMapping.CreateNewMapPoints.OrbMatcher.CheckDistEpipolarLine	0.264935065
Optimizer.GlobalBundleAdjustment.robustHuberKernelDelta	0.242857143
LocalMapping.LocalBundleAdjustment.chi2ErrorThStereo	0.227272727
ORBextractor.nFeatures	0.218181818
Track.Relocalization.narrowedInliers	0.218181818
Tracking.MonocularInitialization.Initializer.ReconstructF.nSimilarTh	0.207792208
Tracking.MonocularInitialization.Initializer.ReconstructH.countInliersTh2	0.192546584
LocalMapping.LocalBundleAdjustment.iterationsWithOutliers	0.177922078
LoopClosing.CorrectLoop.SearchAndFuse.OrbMatcher.LowesRatio	0.164935065
Tracking.MonocularInitialization.Initializer.mMaxIterations	0.161038961
LoopClosing.CorrectLoop.SearchAndFuse.OrbMatcher.Fuse.threshod	0.15974026
Tracking.Track.TrackReferenceKeyFrame.PoseOptimization.robustHuberKernalDeltaStereo	0.155844156
Tracking.Track.TrackReferenceKeyFrame.OrbMatcher.LowesRatio	0.153246753
LocalMapping.KeyFrameCulling.redundantTh	0.153020892
Track.NeedNewFrame.thRefRatioMono	0.141727837
LoopClosing.ComputeSim3.OrbMatcher.SearchBySim3.threshold	0.141558442
LocalMapping.CreateNewMapPoints.sigmaSquared2MultiplierStereo	0.136363636
LoopClosing.KeyFrameDatabase.minScoreToRetainRatio	0.136363636
Tracking.MonocularInitialization.Initializer.ReconstructH.minTriagulated	0.114285714
Track.NeedNewFrame.thRefRatioStereo	0.11038961
Tracking.Track.Relocalization.PnPSolver.SetRansacParameters.minSet	0.102597403
Tracking.MonocularInitialization.Initializer.ReconstructF.countInliersTh1	0.087012987
Tracking.Track.NeedNewFrame.nMaxFrames	0.085714286
Tracking.Track.Relocalization.PnPSolver.SetRansacParameters.th2	0.084415584
Tracking.Track.TrackWithMotionModel.MinMatches	0.077922078
Tracking.Track.TrackReferenceKeyFrame.PoseOptimization.robustHuberKernalDeltaMono	0.076623377
LocalMapping.MapPointCulling.recentlyAddedCount	0.069452287
LoopClosing.DetectLoops.KeyframesPast	0.063636364
LocalMapping.MapPointCulling.cullInGeneral	0.062337662
Tracking.Track.thLostToFast	0.062337662
Tracking.Track.Relocalization.MinMatches	0.055844156
Tracking.Track.Relocalization.secondMinInliers	0.050649351

LocalMapping.CreateNewMapPoints.ratioFactorMultiplier	0.049350649
LocalMapping.LocalBundleAdjustment.thHuberMono	0.038961039
LocalMapping.CreateNewMapPoints.OrbMatcher	0.035064935
LoopClosing.ComputeSim3.OrbMatcher.SearchByBoWMinMatches	0.033766234
LocalMapping.LocalBundleAdjustment.chi2ErrorThMono	0.027272727
LoopClosing.KeyFrameDatabase.numBestCovisbility	0.020779221
Tracking.MonocularInitialization.Initializer.Initialize.RH	0.006493506
Tracking.Track.TrackLocalMap.inlierThres	-0.001298701
Tracking.Track.Relocalization.PnPSolver.SetRansacParameters.maxIterations	-0.003896104
Tracking.MonocularInitialization.Initializer.ReconstructH.countInliersTh1	-0.006493506
LocalMapping.LocalBundleAdjustment.thHuberStereo	-0.01038961
Tracking.MonocularInitialization.OrbMatcher.LowesRatio	-0.014116318
Tracking.MonocularInitialization.Initializer.ReconstructF.minParallax	-0.015584416
LoopClosing.ComputeSim3.OrbMatcher.SearchByProjection.threshold	-0.022077922
LoopClosing.KeyFrameDatabase.minCommonWordsRatio	-0.023376623
Optimizer.OptimizeSim3.iterations	-0.028571429
Tracking.Track.TrackReferenceKeyFrame.PoseOptimization.minInitialCorrespondences	-0.02879729
LocalMapping.MapPointCulling.safeFromCullingCount	-0.035064935
LoopClosing.ComputeSim3.OptimizeSim3NumberInliers	-0.054545455
ORBextractor.scaleFactor	-0.058441558
Tracking.Track.Relocalization.PnPSolver.SetRansacParameters.minInliers	-0.058441558
Tracking.MonocularInitialization.Initializer.ReconstructF.countInliersTh2	-0.063805759
Tracking.Track.TrackReferenceKeyFrame.optminMatches	-0.064935065
Optimizer.OptimizeEssentialGraph.iterations	-0.075324675
LoopClosing.ComputeSim3.Sim3Solver.RansacMaxIterations	-0.076623377
Tracking.Track.TrackLocalMap.inlierThresReloc	-0.076623377
Tracking.Track.Relocalization.SearchByProjection1.orbdist	-0.081818182
LoopClosing.ComputeSim3.Sim3Solver.RansacProbability	-0.084415584
Tracking.MonocularInitialization.Initializer.ReconstructF.checkRT.th	-0.087012987
Tracking.Track.TrackWithMotionModel.SearchByProjection.th2	-0.088311688
Tracking.Track.NeedNewFrame.thCurrentFameTracks	-0.090909091
LoopClosing.ComputeSim3.nTotalMatchTh	-0.098701299
Optimizer.GlobalBundleAdjustment.iterations	-0.1
Tracking.Track.Relocalization.PnPSolver.SetRansacParameters.probablity	-0.125974026
Tracking.MonocularInitialization.Initializer.FindFundamental.thScore	-0.12987013
ORBextractor.minThFAST	-0.133549879
OrbMatcher.TH_HIGH	-0.136363636
LocalMapping.MapPointCulling.foundPercentCorrect	-0.148051948
Tracking.MonocularInitialization.minMatchesBetweenFrames	-0.150649351
Tracking.MonocularInitialization.Initializer.ReconstructF.minTriagulated	-0.155844156
Tracking.MonocularInitialization.Initializer.mSigma	-0.162337662
LocalMapping.CreateNewMapPoints.sigmaSquared2MultiplierMono	-0.167532468
Optimizer.OptimizeEssentialGraph.minFeat	-0.183116883
Tracking.CreateInitialMapMonocular.TrackedMapPoints.minObs	-0.192207792
Tracking.Track.TrackReferenceKeyFrame.nnmatches	-0.192207792
LocalMapping.KeyFrameCulling.nObsTh	-0.216261999
Tracking.Track.TrackWithMotionModel.OrbMatcher.LowesRatio	-0.217391304
Tracking.Track.Relocalization.SearchByProjection2.th	-0.220779221
Tracking.MonocularInitialization.Initializer.FindHomography.th	-0.223376623
Tracking.MonocularInitialization.minTrackedPoints	-0.232467532

LoopClosing.ComputeSim3.OrbMatcher.LowesRatio	-0.244155844
LoopClosing.ComputeSim3.RansacIteration	-0.248051948
Optimizer.OptimizeSim3.th2	-0.255844156
Tracking.CreateInitialMapMonocular.MinTrackedMapPoints	-0.261038961
Tracking.Track.Relocalization.SearchByProjection1.th	-0.280519481
Tracking.Track.NeedNewFrame.referenceKeyFrameNminObs	-0.281761717
Tracking.Track.Relocalization.OrbMatcher2.LowesRatio	-0.297571993
Tracking.MonocularInitialization.Initializer.ReconstructH.checkRT.th	-0.3
LocalMapping.SearchInNeighbours.nn	-0.3
Tracking.Track.TrackReferenceKeyFrame.PoseOptimization.minNumberOfEdges	-0.348051948
LoopClosing.ComputeSim3.Sim3Solver.RansacMinInliers	-0.406493506
Tracking.Track.Relocalization.SearchByProjection2.orbdist	-0.409090909
LocalMapping.SearchInNeighbours.OrbMatcher	-0.423376623
Tracking.MonocularInitialization.OrbMatcher.SearchForInitialization.windowSize	-0.428571429
Tracking.Track.TrackWithMotionModel.MinMapMatches	-0.438961039
Tracking.Track.UpdateLocalMap.UpdateLocalKeyFrames.maxLocal	-0.442857143
Tracking.Track.UpdateLocalMap.UpdateLocalKeyFrames.covisibility	-0.5
Tracking.Track.Relocalization.ransacIterations	-0.532467532
LocalMapping.LocalBundleAdjustment.iterationsWithoutOutliers	-0.576623377
LocalMapping.MapPointCulling.cnThObs	-0.602484472
Keyframe.UpdateConnections.threshold	-0.785714286

Appendix C: Algorithm Training: Configuration Results

iniThFAST	nLevels	GBA.Iterations	MinParallax	minObservations	Fitness Value
24	4	12	1.42	2	0.112039
21	4	22	1.28	2	0.112971
13	12	34	1.22	6	0.113001
7	13	30	1.35	5	0.115626
7	9	20	0.81	6	0.115714
7	9	34	1.6	5	0.116214
21	4	19	1.42	2	0.117535
13	12	34	1.22	6	0.118296
7	13	12	1.39	5	0.118304
21	4	34	1.42	2	0.118989
13	12	22	0.93	6	0.119063
7	13	34	1.22	6	0.119124
23	4	30	1.6	6	0.119314
20	4	20	0.81	3	0.119518
7	9	20	0.81	6	0.119526
13	12	20	1.01	5	0.119607
21	4	34	1.42	2	0.119696
7	9	20	1.39	6	0.1201
13	12	17	0.75	5	0.12029
13	12	30	1.35	5	0.120603
7	13	30	1.6	5	0.120703
7	9	34	1.22	6	0.120871
7	9	20	1.01	5	0.12094
7	13	30	1.35	6	0.121025
7	13	30	0.59	3	0.121183
7	9	20	1.01	5	0.121453
13	12	17	0.81	6	0.121606
7	13	34	1.42	5	0.121628
23	4	19	0.59	3	0.121652
7	13	30	1.22	6	0.121734
21	4	34	1.42	2	0.12175
7	12	34	1.22	6	0.121782
21	4	21	1.28	2	0.121848
7	13	34	1.22	6	0.122094
7	9	34	1.22	6	0.122132
21	4	34	1.42	2	0.12214
13	12	34	1.22	6	0.122173
7	13	30	1.22	5	0.122194
21	4	22	1.01	5	0.12229
21	4	22	0.93	5	0.122291
7	9	20	1.01	5	0.122311
21	4	34	1.42	2	0.122374
21	4	34	1.42	2	0.122432
7	13	30	1.39	6	0.122443
23	4	19	0.81	6	0.122465
7	13	30	1.35	5	0.122472
24	4	36	1.01	5	0.122496
7	4	22	1.28	2	0.122532
7	12	17	0.75	6	0.122624
21	4	34	1.22	5	0.122646
7	13	30	1.35	5	0.122707

(green)	Genetic
(blue)	Grid
(yellow)	Hyperband
(grey)	Random

23	4	19	0.59	3	0.122757
21	4	34	1.39	6	0.122834
7	13	30	1.6	6	0.123059
21	4	34	1.42	2	0.12314
21	4	22	0.93	6	0.123243
24	4	36	1.34	5	0.12332
21	4	34	1.42	2	0.123352
21	4	34	1.42	2	0.123393
14	11	17	1.42	5	0.123409
21	4	12	1.42	2	0.123468
21	4	34	1.42	2	0.123469
21	4	22	1.28	2	0.123699
7	13	12	1.6	5	0.123745
7	13	30	0.81	5	0.123755
21	4	22	1.42	2	0.123772
7	4	22	1.28	2	0.123834
13	12	34	1.22	6	0.123939
24	4	36	0.93	5	0.124052
21	4	22	0.93	6	0.124061
13	12	22	2	6	0.124116
21	4	34	1.34	5	0.124241
21	4	34	1.42	2	0.124307
21	4	12	1.42	2	0.124331
7	13	30	1.6	5	0.12436
7	12	20	1.01	5	0.124465
24	4	22	1.39	6	0.124481
7	13	34	0.93	6	0.12449
7	13	30	0.81	5	0.124513
21	4	34	1.6	2	0.124545
21	4	22	1.28	2	0.124596
13	12	17	0.81	6	0.124628
19	4	21	1.28	2	0.1247
21	4	34	1.34	2	0.124715
24	4	36	1.42	2	0.124773
21	4	22	1.42	2	0.124792
21	4	22	0.93	6	0.124869
21	4	22	1.28	6	0.124923
13	12	17	0.81	6	0.124946
24	4	22	2	5	0.124956
13	12	34	1.22	6	0.124973
21	4	22	1.28	2	0.124975
23	4	22	0.84	5	0.124987
24	4	12	1.42	2	0.12502
7	13	30	1.35	5	0.125026
7	13	20	1.01	5	0.125041
24	4	36	1.34	5	0.125087
21	4	22	1.28	6	0.125099
24	4	34	1.34	5	0.125105
7	11	17	1.42	5	0.125142
21	4	34	1.22	4	0.125146
7	9	20	1.01	5	0.125152
7	13	30	1.42	5	0.125311

7	9	20	0.93	5	0.125323
21	4	34	1.35	5	0.125324
21	4	19	1.6	5	0.125326
21	4	20	1.01	5	0.125361
21	4	22	1.28	2	0.125432
13	4	34	1.42	2	0.126172
21	4	34	1.42	2	0.126182
23	4	30	1.6	6	0.126713
13	12	34	1.22	5	0.127177
7	4	21	1.28	2	0.127856
7	9	34	1.28	6	0.127881
7	13	34	1.39	6	0.128136
7	9	20	1.39	6	0.128221
21	4	34	1.42	2	0.128954
21	4	12	1.39	2	0.128995
21	4	34	1.42	2	0.129314
7	13	30	1.22	6	0.13013
21	4	34	1.22	6	0.130294
13	12	34	1.22	6	0.130396
13	12	20	0.81	6	0.130543
13	12	30	1.35	5	0.130604
7	9	20	0.81	6	0.130636
13	12	17	0.81	5	0.130721
24	4	36	1.28	5	0.130906
21	4	30	1.6	6	0.131054
21	4	34	1.42	2	0.131603
23	4	19	1.39	6	0.132103
7	9	19	1.42	5	0.132443
24	4	12	1.42	2	0.132746
7	13	30	1.35	5	0.133362
7	9	34	1.6	5	0.133486
23	4	19	0.59	3	0.133673
7	13	34	1.22	6	0.134031
21	4	21	1.22	2	0.134435
24	4	30	1.35	5	0.135179
7	4	22	1.01	2	0.135554
7	13	30	0.59	3	0.136207
20	4	30	0.81	3	0.136437
23	4	19	0.59	3	0.13647
20	4	20	0.81	3	0.136493
13	12	30	1.6	5	0.13653
7	9	20	1.01	6	0.137076
21	4	22	0.93	6	0.137108
7	9	34	1.01	5	0.137201
21	4	34	1.42	2	0.137446
23	4	19	0.81	6	0.13832
7	13	30	1.35	6	0.138763
7	13	30	0.59	5	0.138824
7	9	20	1.01	5	0.138859
7	13	30	0.59	6	0.138866
7	9	20	1.01	5	0.139275
13	12	22	0.93	6	0.14036

7	13	30	1.35	5	0.140423
21	4	34	1.42	2	0.140538
7	12	34	1.22	6	0.140661
23	4	17	0.59	3	0.14083
7	13	30	1.6	5	0.140836
7	4	19	1.39	5	0.141291
7	9	20	0.81	6	0.141376
23	4	19	1.35	3	0.141603
23	4	20	1.6	6	0.142018
7	13	30	1.35	5	0.142668
13	12	17	0.75	5	0.142853
7	4	34	1.28	2	0.143048
21	12	34	1.42	2	0.143206
7	4	34	1.42	6	0.143446
7	4	30	1.35	5	0.143805
21	4	34	1.39	6	0.144088
21	13	30	1.42	2	0.144749
7	13	30	1.35	3	0.145167
7	13	30	1.22	5	0.145174
21	13	30	1.22	5	0.145213
13	4	22	1.28	2	0.145256
21	13	34	1.22	5	0.145625
7	4	22	0.93	5	0.146802
7	13	36	1.01	5	0.146976
21	4	22	1.22	5	0.147115
7	13	30	0.81	6	0.147519
13	12	19	0.81	6	0.147674
7	9	20	0.81	6	0.148419
7	13	30	1.42	2	0.149713
7	9	20	1.22	6	0.150687
6	4	10	0.9875	1.5	0.121416
6	4	10	1.55	2.4	0.121999
6	4	10.75	0.9125	3.1875	0.122367
6	4	10.75	0.575	2.9625	0.122899
6	4	10	0.8	1.5	0.122966
6	4	10.75	0.9125	1.95	0.124493
6	4	10	0.9125	2.175	0.124681
6	4	10	1.325	2.4	0.125122
6	4	10	0.9875	1.6125	0.125414
6	4	10	1.8125	1.8375	0.126661
6	4	10	1.1375	1.6125	0.126791
6	4	10	1.325	2.85	0.127336
6	4	10.75	0.5	2.0625	0.12765
6	4	10	1.2875	2.175	0.127734
6	4	10	0.7625	1.6125	0.128179
6	4	10	1.1	4.2	0.128221
6	4	10.75	0.725	2.175	0.128429
6	4	10	1.4	3.8625	0.128616
6	4	10	1.7375	1.8375	0.128787
6	4	10	0.5375	2.0625	0.128916
6	4	10.75	0.9125	2.2875	0.128995
6	4	10	0.65	2.0625	0.12911

6	4	10	1.6625	5.2125	0.129188
6	4	10	1.6625	1.8375	0.129199
6	4	10	1.3625	2.9625	0.129225
6	4	10.75	0.875	2.0625	0.129255
6	4	10	1.25	2.9625	0.129432
6	4	10	1.325	1.8375	0.129475
6	4	10	1.5875	2.4	0.129636
6	4	10	1.55	1.8375	0.130026
6	4	10	0.9875	2.175	0.13003
6	4	10	1.25	2.85	0.130047
6	4	10.75	0.5375	1.95	0.130126
6	4	10	0.7625	2.175	0.130169
6	4	10	1.6625	4.0875	0.130212
6	4	10.75	0.8375	3.75	0.130219
6	4	10	0.8	1.6125	0.13048
6	4	10	1.55	2.625	0.130491
6	4	10	1.1375	2.2875	0.130609
6	4	10	1.3625	3.1875	0.130749
6	4	10	0.95	1.5	0.130826
6	4	10	1.025	1.95	0.130831
6	4	10.75	0.8375	3.525	0.130923
6	4	10	0.6125	1.6125	0.13093
6	4	10	0.5	4.3125	0.131059
6	4	10	1.85	2.2875	0.131068
6	4	10	1.85	5.55	0.131224
6	4	10	0.5	3.075	0.131309
6	4	10	0.5375	1.8375	0.131432
6	4	10	1.85	5.6625	0.131446
6	4	10	1.175	5.1	0.131451
6	4	10	1.925	1.725	0.131466
6	4	10	0.9125	1.8375	0.131526
6	4	10	1.7	1.6125	0.131771
6	4	10.75	0.5375	3.75	0.131938
6	4	10	1.6625	2.2875	0.132079
6	4	10	1.55	2.0625	0.1321
6	4	10	1.5875	1.8375	0.132176
6	4	10	0.5	4.0875	0.132246
6	4	10.75	0.575	2.0625	0.132305
6	4	10	0.575	3.8625	0.132371
6	4	10	1.5875	1.6125	0.132384
6	4	10	1.85	1.5	0.132427
6	4	10	1.175	3.525	0.132429
6	4	10	1.25	5.775	0.132463
6	4	10	0.95	2.0625	0.132492
6	4	10.75	0.725	1.8375	0.132501
6	4	10.75	0.6875	1.5	0.132561
6	4	10	0.5375	5.2125	0.132655
6	4	10	1.5125	1.725	0.132707
6	4	10.75	0.5375	2.7375	0.132707
6	4	10.75	0.8	2.4	0.132823
6	4	10	1.4375	4.2	0.13297
6	4	10	0.95	2.9625	0.133021

6	4	10.75	0.8	2.85	0.13311
6	4	10.75	0.575	2.175	0.133152
6	4	10	1.6625	3.6375	0.133163
6	4	10	1.85	2.0625	0.133303
6	4	10	0.7625	5.1	0.133341
6	4	10	1.925	4.425	0.133391
6	4	10.75	0.725	3.075	0.133409
6	4	10	0.5375	1.6125	0.13347
6	4	10	0.7625	2.625	0.133498
6	4	10.75	0.5	2.2875	0.133548
6	4	10	1.8875	2.7375	0.133572
6	4	10	0.6125	2.9625	0.133594
6	4	10	1.3625	3.75	0.133696
6	4	10	0.8	1.95	0.13373
6	4	10	1.9625	4.7625	0.133759
6	4	10	1.775	1.6125	0.13393
6	4	10	0.725	1.5	0.133975
6	4	10	1.55	5.4375	0.133983
6	4	10	1.7375	2.625	0.134086
6	4	10	1.7375	3.3	0.134086
6	4	10.75	0.5375	5.8875	0.134294
6	4	10	0.65	1.5	0.1343
6	4	10	1.2125	1.95	0.1343
6	4	10.75	0.8	1.95	0.1343
6	4	10	1.9625	1.95	0.134345
6	4	10.75	0.7625	1.8375	0.134411
6	4	10	0.5375	2.9625	0.134429
6	4	10	0.7625	1.8375	0.134435
6	4	10	1.6625	2.0625	0.134464
6	4	10.75	0.7625	3.4125	0.134467
6	4	10	0.5	2.625	0.134483
6	4	10	1.475	3.3	0.134545
6	4	10	1.9625	2.2875	0.134549
6	4	10	1.2125	5.4375	0.134578
6	4	10	1.0625	2.2875	0.134582
6	4	10	1.7375	2.9625	0.134668
6	4	10.75	0.6875	3.075	0.134685
6	4	10.75	0.725	1.725	0.134735
6	4	10	1.25	4.875	0.134773
6	4	10.75	0.7625	3.1875	0.134774
6	4	10	0.5	1.6125	0.134803
6	4	10	1.1375	5.55	0.134876
6	4	10	1.925	3.6375	0.134894
6	4	10	1.7	1.5	0.134944
6	4	10	0.875	2.2875	0.134989
6	4	10	1.3625	5.775	0.134999
6	4	10.75	0.875	5.4375	0.135071
6	4	10.75	0.8375	1.725	0.135079
6	4	10	1.0625	4.3125	0.135096
6	4	10.75	0.5	3.8625	0.135191
6	4	10	1.3625	3.525	0.135241
6	4	10	0.9875	6.1125	0.135247

6	4	10	1.1	2.625	0.135249
6	4	10	1.85	2.625	0.135249
6	4	10	0.725	3.075	0.135332
6	4	10	1.8125	5.6625	0.135336
6	4	10.75	0.8	5.325	0.135356
6	4	10.75	0.5375	2.5125	0.135359
6	4	10	1.7	6	0.1354
6	4	10.75	0.8	3.4125	0.135417
6	4	10	0.8	1.725	0.135577
6	4	10.75	0.65	1.8375	0.135582
6	4	10.75	0.8375	2.7375	0.135687
6	4	10	0.9875	2.5125	0.135739
6	4	10	0.6125	4.7625	0.135791
6	4	10	0.8	5.4375	0.135839
6	4	10	0.575	5.775	0.135948
6	4	10	1.775	3.1875	0.135955
6	4	10	0.7625	2.9625	0.135967
6	4	10	1.5125	4.0875	0.136007
6	4	10	1.325	4.5375	0.136026
6	4	10	1.5875	2.5125	0.136088
6	4	10	1.2875	3.75	0.136117
6	4	10	0.875	2.0625	0.136307
6	4	10.75	0.9125	2.5125	0.136337
6	4	10	0.9125	3.3	0.136384
6	4	10	1.2125	5.325	0.136384
6	4	10	1.55	3.975	0.136387
6	4	10	1.4	2.175	0.136401
6	4	10	0.65	2.2875	0.136482
6	4	10	1.8125	4.65	0.136534
6	4	10.75	0.5	1.95	0.136542
6	4	10	1.4375	3.075	0.136549
6	4	10	0.575	4.875	0.136615
6	4	10	1.2875	2.7375	0.136644
6	4	10.75	0.6875	3.1875	0.136689
6	4	10	0.725	2.0625	0.136736
6	4	10	0.65	2.175	0.136795
6	4	10.75	0.7625	2.625	0.136795
6	4	10	1.325	1.6125	0.136844
6	4	10.75	0.5375	5.775	0.13685
6	4	10.75	0.65	2.7375	0.136899
6	4	10	0.95	5.775	0.137021
6	4	10	0.9125	2.625	0.137043
6	4	10.75	0.875	2.625	0.137075
6	4	10	0.575	1.725	0.137106
6	4	10	1.6625	4.65	0.137116
6	4	10	1.175	3.4125	0.137122
6	4	10.75	0.8375	2.85	0.137167
6	4	10	1.475	4.0875	0.137168
6	4	10	1.2125	4.0875	0.137178
6	4	10	0.65	2.5125	0.137202
6	4	10	1.925	2.9625	0.137203
6	4	10.75	0.6125	3.6375	0.137289

6	4	10	1.775	2.9625	0.137291
6	4	10	1.1375	3.8625	0.137335
6	4	10	1.8875	2.5125	0.13736
6	4	10	1.5125	5.1	0.137394
6	4	10	1.8875	6	0.137419
6	4	10	0.7625	5.6625	0.13742
6	4	10	1.175	3.6375	0.137447
6	4	10	1.9625	5.2125	0.137482
6	4	10	0.7625	4.2	0.137485
6	4	10	0.8375	4.3125	0.137504
6	4	10	1.25	1.5	0.137525
6	4	10.75	0.875	5.8875	0.137545
6	4	10	0.5	5.2125	0.137593
6	4	10.75	0.6125	2.625	0.137596
6	4	10	1.8125	2.625	0.137615
6	4	10	1.5125	3.4125	0.137672
6	4	10	1.7	2.4	0.137696
6	4	10	1.0625	6.1125	0.13775
6	4	10.75	0.8375	4.425	0.137755
6	4	10	1.8125	4.2	0.13781
6	4	10.75	0.6875	2.175	0.137845
6	4	10	1.0625	3.6375	0.137886
6	4	10	1.1375	3.1875	0.137886
6	4	10	1.925	3.3	0.137897
6	4	10	1.925	2.0625	0.137913
6	4	10	0.8375	1.6125	0.13792
6	4	10	1.8125	4.0875	0.137938
6	4	10	0.575	3.6375	0.13804
6	4	10	0.6125	5.55	0.138047
6	4	10.75	0.9125	1.5	0.138053
6	4	10	0.95	4.0875	0.138054
6	4	10	1.3625	5.1	0.138136
6	4	10	1.85	2.85	0.138165
6	4	10	1.8125	2.9625	0.138201
6	4	10	1.175	5.55	0.138215
6	4	10	1.5875	1.5	0.138222
6	4	10	0.5	2.0625	0.138264
6	4	10	1.7	5.1	0.138361
6	4	10	1.1375	4.425	0.13838
6	4	10.75	0.6875	4.9875	0.138396
6	4	10	1.25	3.4125	0.138403
6	4	10	1.55	3.075	0.138405
6	4	10	1.7	3.1875	0.138408
6	4	10	0.95	2.625	0.138416
6	4	10.75	0.575	6	0.13846
6	4	10.75	0.575	2.85	0.138476
6	4	10.75	0.7625	4.3125	0.138501
6	4	10	1.325	6	0.138514
6	4	10	1.4375	3.6375	0.138524
6	4	10	1.1	3.4125	0.138581
6	4	10	1.925	1.5	0.138588
6	4	10	1.7375	1.95	0.138591

6	4	10	1.7	2.175	0.138596
6	4	10	0.8	4.0875	0.138615
6	4	10	1.5125	4.425	0.138627
6	4	10	1.5125	2.625	0.138681
6	4	10	1.2125	2.9625	0.138699
6	4	10	1.475	4.5375	0.13874
6	4	10	1.2125	5.6625	0.138767
6	4	10.75	0.875	5.55	0.1388
6	4	10	0.6125	2.7375	0.138825
6	4	10	1.2125	2.7375	0.13883
6	4	10	1.1375	2.4	0.138834
6	4	10	0.7625	6.1125	0.138844
6	4	10	1.2125	3.525	0.138854
6	4	10.75	0.5	3.975	0.138854
6	4	10	1.925	2.2875	0.138878
6	4	10	1.0625	4.425	0.138897
6	4	10	1.3625	5.325	0.138898
6	4	10.75	0.6875	3.8625	0.13893
6	4	10	1.7375	6	0.138979
6	4	10	1.025	1.8375	0.139
6	4	10	0.8	2.4	0.139101
6	4	10.75	0.9125	1.725	0.139105
6	4	10	0.6125	2.5125	0.139118
6	4	10	1.2125	3.3	0.139136
6	4	10	1.775	5.55	0.139226
6	4	10	1.7375	4.65	0.139288
6	4	10	1.7	5.2125	0.139298
6	4	10.75	0.65	2.2875	0.13935
6	4	10	0.725	5.2125	0.139351
6	4	10	0.9875	5.1	0.139389
6	4	10	1.55	5.2125	0.13939
6	4	10	1.1375	4.0875	0.139402
6	4	10.75	0.5	4.2	0.139411
6	4	10.75	0.875	1.8375	0.139447
6	4	10	0.875	5.775	0.139486
6	4	10	0.8375	3.6375	0.139503
6	4	10	0.95	1.8375	0.139551
6	4	10	1.325	4.65	0.139584
6	4	10	0.8375	2.4	0.139596
6	4	10	1.2875	4.7625	0.1396
6	4	10	0.5	6	0.139614
6	4	10.75	0.6125	5.8875	0.139665
6	4	10.75	0.5	4.65	0.139787
6	4	10.75	0.875	1.95	0.13983
6	4	10	1.175	1.6125	0.139875
6	4	10.75	0.8375	4.3125	0.139891
6	4	10	1.6625	5.1	0.13992
6	4	10	1.3625	2.4	0.139943
6	4	10	1.9625	5.6625	0.139968
6	4	10	0.9875	3.525	0.140021
6	4	10	0.6875	1.95	0.140088
6	4	10	0.6125	3.8625	0.1401

6	4	10	0.6875	5.2125	0.14012
6	4	10	1.025	2.175	0.140151
6	4	10.75	0.7625	5.8875	0.140228
6	4	10	1.1	3.075	0.140242
6	4	10	1.1	2.85	0.140284
6	4	10	0.95	2.5125	0.140323
6	4	10.75	0.6875	2.625	0.14037
6	4	10	0.875	5.8875	0.140384
6	4	10.75	0.8	1.725	0.140414
6	4	10	0.5	2.2875	0.140428
6	4	10	0.8	3.4125	0.140502
6	4	10	1.625	5.775	0.140504
6	4	10	0.5	5.325	0.140511
6	4	10	1.5125	3.1875	0.140538
6	4	10	1.1	6.1125	0.140585
6	4	10	1.6625	5.6625	0.140678
6	4	10	1.85	2.4	0.140697
6	4	10.75	0.875	2.175	0.140839
6	4	10.75	0.65	6.1125	0.140845
6	4	10	0.9875	4.3125	0.14087
6	4	10	1.625	1.5	0.140889
6	4	10	0.95	2.7375	0.140891
6	4	10	1.925	3.075	0.140896
6	4	10	1.2875	6.1125	0.14092
6	4	10.75	0.7625	3.6375	0.140939
6	4	10.75	0.65	4.5375	0.140953
11	6	26	1.61	3	0.134472375
13	8	30	0.95	4	0.1358375
21	11	23	1.18	5	0.137905
15	4	24	1.97	6	0.1389325
9	11	35	1.99	5	0.126316
18	10	19	1.36	4	0.134312
14	14	34	1.85	3	0.136593
16	7	18	1.91	3	0.138604
14	14	30	1.24	2	0.140022
10	7	16	0.74	3	0.1405048
16	14	37	0.6	3	0.1350195
19	8	25	1.94	2	0.137673875
23	12	22	1.18	3	0.137691
19	6	38	0.82	2	0.126949
12	11	32	1.29	6	0.128273
23	4	21	1.29	6	0.12978
15	5	16	1.48	2	0.131693
21	4	25	1.58	4	0.133144
15	5	15	1.2	2	0.133446
6	8	31	0.78	5	0.133577
23	6	39	1.62	3	0.133881
20	5	31	1.27	3	0.134104
16	12	17	0.61	5	0.134834
17	14	38	1.88	6	0.134846
11	4	24	1.42	3	0.134952
10	14	22	1.08	3	0.13515

8	11	22	1.36	6	0.136214
18	8	32	0.82	3	0.136646
20	6	26	1.59	3	0.137087
14	4	39	0.89	2	0.137257
10	9	31	1.28	3	0.137308
10	6	39	1.65	2	0.137542
15	13	36	1.52	3	0.137602
6	11	16	1.62	3	0.137993
7	6	29	1.07	2	0.138196
10	9	24	1.37	2	0.138556
6	12	16	0.55	2	0.138677
14	7	37	1.27	2	0.138737
13	6	10	1.25	3	0.138785
10	6	12	1.02	2	0.139117
13	14	40	1.93	3	0.139388
19	4	39	1.72	5	0.139512
7	5	36	0.53	3	0.139559
8	6	10	1.06	2	0.139761
19	14	35	1.8	3	0.139812
11	12	15	1.8	3	0.139837
13	14	18	1.24	6	0.139928
12	8	15	1	2	0.139974
8	10	24	1.35	5	0.140094
23	6	28	1.81	3	0.140158
21	5	40	1.26	5	0.140261
13	4	19	1.46	3	0.141052
20	11	17	1.5	5	0.141152
13	10	38	0.57	5	0.141766
8	11	21	0.71	2	0.141841
9	4	21	1.49	2	0.142348
12	14	34	0.83	5	0.142408
22	5	29	1.13	3	0.142784
14	4	31	1.02	3	0.143101
24	9	38	0.78	4	0.143163
13	12	20	1.54	5	0.143484
23	7	22	1.19	3	0.143545
7	6	37	0.75	6	0.143558
13	6	28	1.2	2	0.143628
21	7	22	0.79	4	0.143728
17	5	36	1.73	2	0.143795
12	11	15	1.08	4	0.143921
11	11	18	0.69	2	0.144021
10	14	30	0.77	2	0.144141
9	7	31	1.2	3	0.144777
10	10	13	1.23	5	0.145086
13	13	15	0.87	2	0.145408
8	4	20	1.37	2	0.145536
10	8	25	1.57	2	0.14572
12	13	30	1.33	6	0.146268
7	6	18	1.23	5	0.146273
19	4	28	1.95	4	0.146352
20	11	14	1.39	5	0.146739

11	9	10	1.82	4	0.146756
8	13	10	0.87	6	0.146772
10	8	32	0.52	2	0.146803
12	5	14	0.97	6	0.147034
16	10	35	0.78	2	0.147389
12	5	22	1.13	2	0.147427
6	6	35	1.64	2	0.147492
13	5	13	0.73	2	0.147559
15	6	26	1.89	4	0.147658
20	5	30	1.52	6	0.147686
11	4	31	0.71	5	0.147777
15	7	21	1.93	2	0.147813
8	14	19	1.98	4	0.148114
9	5	38	1.45	2	0.148194
20	11	33	1.71	2	0.148291
20	7	28	1.38	5	0.148945
7	14	13	1.21	3	0.14924
16	7	26	1.47	3	0.149281
10	9	38	0.93	3	0.149301
21	11	37	1.7	2	0.149697
18	6	15	1.79	2	0.150471
20	7	36	0.56	3	0.150545
6	11	15	1.32	6	0.150664
13	9	36	0.55	3	0.150726
10	11	29	1.9	2	0.150974
14	6	12	1.26	4	0.151107
7	10	30	1.41	5	0.151109
10	7	10	1.9	2	0.151125
7	8	34	1.14	2	0.152231
8	7	17	1.74	3	0.152378
16	6	39	0.61	5	0.153063
21	12	32	0.77	6	0.153205
15	4	22	1.67	5	0.153231
23	8	26	1.13	3	0.153275
14	5	24	0.88	6	0.153333
13	10	24	0.68	5	0.153404
14	13	36	0.51	6	0.153413
14	13	11	1.25	3	0.153581
20	8	32	1.19	4	0.153818
19	8	21	1.68	3	0.153843

Appendix D: Selected Trained Configurations

iniThFAST	nLevels	GBA.Iterations	MinParallax	minObservations
24	4	12	1.42	2
21	4	22	1.28	2
13	12	34	1.22	6
7	13	30	1.35	5
7	9	20	0.81	6
7	9	34	1.6	5
21	4	19	1.42	2
13	12	34	1.22	6
7	13	12	1.39	5
21	4	34	1.42	2
13	12	22	0.93	6
7	13	34	1.22	6
23	4	30	1.6	6
20	4	20	0.81	3
7	9	20	0.81	6
13	12	20	1.01	5
21	4	34	1.42	2
7	9	20	1.39	6
13	12	17	0.75	5
13	12	30	1.35	5
7	13	30	1.6	5
7	9	34	1.22	6
7	9	20	1.01	5
7	13	30	1.35	6
7	13	30	0.59	3
7	9	20	1.01	5
13	12	17	0.81	6
7	13	34	1.42	5
23	4	19	0.59	3
7	13	30	1.22	6
21	4	34	1.42	2
7	12	34	1.22	6
21	4	21	1.28	2
7	13	34	1.22	6
7	9	34	1.22	6
21	4	34	1.42	2
13	12	34	1.22	6
7	13	30	1.22	5
21	4	22	1.01	5
21	4	22	0.93	5
7	9	20	1.01	5
21	4	34	1.42	2
21	4	34	1.42	2

- Genetic
- Grid
- Hyperband
- Random

7	13	30	1.39	6
23	4	19	0.81	6
7	13	30	1.35	5
24	4	36	1.01	5
7	4	22	1.28	2
7	12	17	0.75	6
21	4	34	1.22	5
7	13	30	1.35	5
23	4	19	0.59	3
21	4	34	1.39	6
7	13	30	1.6	6
21	4	34	1.42	2
21	4	22	0.93	6
24	4	36	1.34	5
21	4	34	1.42	2
21	4	34	1.42	2
14	11	17	1.42	5
21	4	12	1.42	2
21	4	34	1.42	2
21	4	22	1.28	2
7	13	12	1.6	5
7	13	30	0.81	5
21	4	22	1.42	2
7	4	22	1.28	2
13	12	34	1.22	6
24	4	36	0.93	5
21	4	22	0.93	6
13	12	22	2	6
21	4	34	1.34	5
21	4	34	1.42	2
21	4	12	1.42	2
7	13	30	1.6	5
7	12	20	1.01	5
24	4	22	1.39	6
7	13	34	0.93	6
7	13	30	0.81	5
21	4	34	1.6	2
21	4	22	1.28	2
13	12	17	0.81	6
19	4	21	1.28	2
21	4	34	1.34	2
24	4	36	1.42	2
21	4	22	1.42	2
21	4	22	0.93	6

21	4	22	1.28	6
13	12	17	0.81	6
24	4	22	2	5
13	12	34	1.22	6
21	4	22	1.28	2
23	4	22	0.84	5
24	4	12	1.42	2
7	13	30	1.35	5
7	13	20	1.01	5
24	4	36	1.34	5
21	4	22	1.28	6
24	4	34	1.34	5
7	11	17	1.42	5
21	4	34	1.22	4
7	9	20	1.01	5
7	13	30	1.42	5
7	9	20	0.93	5
21	4	34	1.35	5
21	4	19	1.6	5
21	4	20	1.01	5
21	4	22	1.28	2
13	4	34	1.42	2
21	4	34	1.42	2
23	4	30	1.6	6
13	12	34	1.22	5
7	4	21	1.28	2
7	9	34	1.28	6
7	13	34	1.39	6
7	9	20	1.39	6
21	4	34	1.42	2
21	4	12	1.39	2
21	4	34	1.42	2
7	13	30	1.22	6
21	4	34	1.22	6
13	12	34	1.22	6
13	12	20	0.81	6
13	12	30	1.35	5
7	9	20	0.81	6
13	12	17	0.81	5
24	4	36	1.28	5
21	4	30	1.6	6
21	4	34	1.42	2
23	4	19	1.39	6
7	9	19	1.42	5

24	4	12	1.42	2
7	13	30	1.35	5
7	9	34	1.6	5
23	4	19	0.59	3
7	13	34	1.22	6
21	4	21	1.22	2
24	4	30	1.35	5
7	4	22	1.01	2
7	13	30	0.59	3
20	4	30	0.81	3
23	4	19	0.59	3
20	4	20	0.81	3
13	12	30	1.6	5
7	9	20	1.01	6
21	4	22	0.93	6
7	9	34	1.01	5
21	4	34	1.42	2
23	4	19	0.81	6
7	13	30	1.35	6
7	13	30	0.59	5
7	9	20	1.01	5
7	13	30	0.59	6
7	9	20	1.01	5
13	12	22	0.93	6
7	13	30	1.35	5
21	4	34	1.42	2
7	12	34	1.22	6
23	4	17	0.59	3
7	13	30	1.6	5
7	4	19	1.39	5
7	9	20	0.81	6
23	4	19	1.35	3
23	4	20	1.6	6
7	13	30	1.35	5
13	12	17	0.75	5
7	4	34	1.28	2
21	12	34	1.42	2
7	4	34	1.42	6
7	4	30	1.35	5
21	4	34	1.39	6
21	13	30	1.42	2
7	13	30	1.35	3
7	13	30	1.22	5
21	13	30	1.22	5

13	4	22	1.28	2
21	13	34	1.22	5
7	4	22	0.93	5
7	13	36	1.01	5
21	4	22	1.22	5
7	13	30	0.81	6
13	12	19	0.81	6
7	9	20	0.81	6
7	13	30	1.42	2
7	9	20	1.22	6
6	4	10	0.9875	1.5
6	4	10	1.55	2.4
6	4	10.75	0.9125	3.1875
6	4	10.75	0.575	2.9625
6	4	10	0.8	1.5
6	4	10.75	0.9125	1.95
6	4	10	0.9125	2.175
6	4	10	1.325	2.4
6	4	10	0.9875	1.6125
6	4	10	1.8125	1.8375
6	4	10	1.1375	1.6125
6	4	10	1.325	2.85
6	4	10.75	0.5	2.0625
6	4	10	1.2875	2.175
6	4	10	0.7625	1.6125
6	4	10	1.1	4.2
6	4	10.75	0.725	2.175
6	4	10	1.4	3.8625
6	4	10	1.7375	1.8375
6	4	10	0.5375	2.0625
6	4	10.75	0.9125	2.2875
6	4	10	0.65	2.0625
6	4	10	1.6625	5.2125
6	4	10	1.6625	1.8375
6	4	10	1.3625	2.9625
6	4	10.75	0.875	2.0625
6	4	10	1.25	2.9625
6	4	10	1.325	1.8375
6	4	10	1.5875	2.4
6	4	10	1.55	1.8375
6	4	10	0.9875	2.175
6	4	10	1.25	2.85
6	4	10.75	0.5375	1.95
6	4	10	0.7625	2.175

6	4	10	1.6625	4.0875
6	4	10.75	0.8375	3.75
6	4	10	0.8	1.6125
6	4	10	1.55	2.625
6	4	10	1.1375	2.2875
6	4	10	1.3625	3.1875
6	4	10	0.95	1.5
6	4	10	1.025	1.95
6	4	10.75	0.8375	3.525
6	4	10	0.6125	1.6125
6	4	10	0.5	4.3125
6	4	10	1.85	2.2875
6	4	10	1.85	5.55
6	4	10	0.5	3.075
6	4	10	0.5375	1.8375
6	4	10	1.85	5.6625
6	4	10	1.175	5.1
6	4	10	1.925	1.725
6	4	10	0.9125	1.8375
6	4	10	1.7	1.6125
6	4	10.75	0.5375	3.75
6	4	10	1.6625	2.2875
6	4	10	1.55	2.0625
6	4	10	1.5875	1.8375
6	4	10	0.5	4.0875
6	4	10.75	0.575	2.0625
6	4	10	0.575	3.8625
6	4	10	1.5875	1.6125
6	4	10	1.85	1.5
6	4	10	1.175	3.525
6	4	10	1.25	5.775
6	4	10	0.95	2.0625
6	4	10.75	0.725	1.8375
6	4	10.75	0.6875	1.5
6	4	10	0.5375	5.2125
6	4	10	1.5125	1.725
6	4	10.75	0.5375	2.7375
6	4	10.75	0.8	2.4
6	4	10	1.4375	4.2
6	4	10	0.95	2.9625
6	4	10.75	0.8	2.85
6	4	10.75	0.575	2.175
6	4	10	1.6625	3.6375
6	4	10	1.85	2.0625

6	4	10	0.7625	5.1
6	4	10	1.925	4.425
6	4	10.75	0.725	3.075
6	4	10	0.5375	1.6125
6	4	10	0.7625	2.625
6	4	10.75	0.5	2.2875
6	4	10	1.8875	2.7375
6	4	10	0.6125	2.9625
6	4	10	1.3625	3.75
6	4	10	0.8	1.95
6	4	10	1.9625	4.7625
6	4	10	1.775	1.6125
6	4	10	0.725	1.5
6	4	10	1.55	5.4375
6	4	10	1.7375	2.625
6	4	10	1.7375	3.3
6	4	10.75	0.5375	5.8875
6	4	10	0.65	1.5
6	4	10	1.2125	1.95
6	4	10.75	0.8	1.95
6	4	10	1.9625	1.95
6	4	10.75	0.7625	1.8375
6	4	10	0.5375	2.9625
6	4	10	0.7625	1.8375
6	4	10	1.6625	2.0625
6	4	10.75	0.7625	3.4125
6	4	10	0.5	2.625
6	4	10	1.475	3.3
6	4	10	1.9625	2.2875
6	4	10	1.2125	5.4375
6	4	10	1.0625	2.2875
6	4	10	1.7375	2.9625
6	4	10.75	0.6875	3.075
6	4	10.75	0.725	1.725
6	4	10	1.25	4.875
6	4	10.75	0.7625	3.1875
6	4	10	0.5	1.6125
6	4	10	1.1375	5.55
6	4	10	1.925	3.6375
6	4	10	1.7	1.5
6	4	10	0.875	2.2875
6	4	10	1.3625	5.775
6	4	10.75	0.875	5.4375
6	4	10.75	0.8375	1.725

6	4	10	1.0625	4.3125
6	4	10.75	0.5	3.8625
6	4	10	1.3625	3.525
6	4	10	0.9875	6.1125
6	4	10	1.1	2.625
6	4	10	1.85	2.625
6	4	10	0.725	3.075
6	4	10	1.8125	5.6625
6	4	10.75	0.8	5.325
6	4	10.75	0.5375	2.5125
6	4	10	1.7	6
6	4	10.75	0.8	3.4125
6	4	10	0.8	1.725
6	4	10.75	0.65	1.8375
6	4	10.75	0.8375	2.7375
6	4	10	0.9875	2.5125
6	4	10	0.6125	4.7625
6	4	10	0.8	5.4375
6	4	10	0.575	5.775
6	4	10	1.775	3.1875
6	4	10	0.7625	2.9625
6	4	10	1.5125	4.0875
6	4	10	1.325	4.5375
6	4	10	1.5875	2.5125
6	4	10	1.2875	3.75
6	4	10	0.875	2.0625
6	4	10.75	0.9125	2.5125
6	4	10	0.9125	3.3
6	4	10	1.2125	5.325
6	4	10	1.55	3.975
6	4	10	1.4	2.175
6	4	10	0.65	2.2875
6	4	10	1.8125	4.65
6	4	10.75	0.5	1.95
6	4	10	1.4375	3.075
6	4	10	0.575	4.875
6	4	10	1.2875	2.7375
6	4	10.75	0.6875	3.1875
6	4	10	0.725	2.0625
6	4	10	0.65	2.175
6	4	10.75	0.7625	2.625
6	4	10	1.325	1.6125
6	4	10.75	0.5375	5.775
6	4	10.75	0.65	2.7375

6	4	10	0.95	5.775
6	4	10	0.9125	2.625
6	4	10.75	0.875	2.625
6	4	10	0.575	1.725
6	4	10	1.6625	4.65
6	4	10	1.175	3.4125
6	4	10.75	0.8375	2.85
6	4	10	1.475	4.0875
6	4	10	1.2125	4.0875
6	4	10	0.65	2.5125
6	4	10	1.925	2.9625
6	4	10.75	0.6125	3.6375
6	4	10	1.775	2.9625
6	4	10	1.1375	3.8625
6	4	10	1.8875	2.5125
6	4	10	1.5125	5.1
6	4	10	1.8875	6
6	4	10	0.7625	5.6625
6	4	10	1.175	3.6375
6	4	10	1.9625	5.2125
6	4	10	0.7625	4.2
6	4	10	0.8375	4.3125
6	4	10	1.25	1.5
6	4	10.75	0.875	5.8875
6	4	10	0.5	5.2125
6	4	10.75	0.6125	2.625
6	4	10	1.8125	2.625
6	4	10	1.5125	3.4125
6	4	10	1.7	2.4
6	4	10	1.0625	6.1125
6	4	10.75	0.8375	4.425
6	4	10	1.8125	4.2
6	4	10.75	0.6875	2.175
6	4	10	1.0625	3.6375
6	4	10	1.1375	3.1875
6	4	10	1.925	3.3
6	4	10	1.925	2.0625
6	4	10	0.8375	1.6125
6	4	10	1.8125	4.0875
6	4	10	0.575	3.6375
6	4	10	0.6125	5.55
6	4	10.75	0.9125	1.5
6	4	10	0.95	4.0875
6	4	10	1.3625	5.1

6	4	10	1.85	2.85
6	4	10	1.8125	2.9625
6	4	10	1.175	5.55
6	4	10	1.5875	1.5
6	4	10	0.5	2.0625
6	4	10	1.7	5.1
6	4	10	1.1375	4.425
6	4	10.75	0.6875	4.9875
6	4	10	1.25	3.4125
6	4	10	1.55	3.075
6	4	10	1.7	3.1875
6	4	10	0.95	2.625
6	4	10.75	0.575	6
6	4	10.75	0.575	2.85
6	4	10.75	0.7625	4.3125
6	4	10	1.325	6
6	4	10	1.4375	3.6375
6	4	10	1.1	3.4125
6	4	10	1.925	1.5
6	4	10	1.7375	1.95
6	4	10	1.7	2.175
6	4	10	0.8	4.0875
6	4	10	1.5125	4.425
6	4	10	1.5125	2.625
6	4	10	1.2125	2.9625
6	4	10	1.475	4.5375
6	4	10	1.2125	5.6625
6	4	10.75	0.875	5.55
6	4	10	0.6125	2.7375
6	4	10	1.2125	2.7375
6	4	10	1.1375	2.4
6	4	10	0.7625	6.1125
6	4	10	1.2125	3.525
6	4	10.75	0.5	3.975
6	4	10	1.925	2.2875
6	4	10	1.0625	4.425
6	4	10	1.3625	5.325
6	4	10.75	0.6875	3.8625
6	4	10	1.7375	6
6	4	10	1.025	1.8375
6	4	10	0.8	2.4
6	4	10.75	0.9125	1.725
6	4	10	0.6125	2.5125
6	4	10	1.2125	3.3

6	4	10	1.775	5.55
6	4	10	1.7375	4.65
6	4	10	1.7	5.2125
6	4	10.75	0.65	2.2875
6	4	10	0.725	5.2125
6	4	10	0.9875	5.1
6	4	10	1.55	5.2125
6	4	10	1.1375	4.0875
6	4	10.75	0.5	4.2
6	4	10.75	0.875	1.8375
6	4	10	0.875	5.775
6	4	10	0.8375	3.6375
6	4	10	0.95	1.8375
6	4	10	1.325	4.65
6	4	10	0.8375	2.4
6	4	10	1.2875	4.7625
6	4	10	0.5	6
6	4	10.75	0.6125	5.8875
6	4	10.75	0.5	4.65
6	4	10.75	0.875	1.95
6	4	10	1.175	1.6125
6	4	10.75	0.8375	4.3125
6	4	10	1.6625	5.1
6	4	10	1.3625	2.4
6	4	10	1.9625	5.6625
6	4	10	0.9875	3.525
6	4	10	0.6875	1.95
6	4	10	0.6125	3.8625
6	4	10	0.6875	5.2125
6	4	10	1.025	2.175
6	4	10.75	0.7625	5.8875
6	4	10	1.1	3.075
6	4	10	1.1	2.85
6	4	10	0.95	2.5125
6	4	10.75	0.6875	2.625
6	4	10	0.875	5.8875
6	4	10.75	0.8	1.725
6	4	10	0.5	2.2875
6	4	10	0.8	3.4125
6	4	10	1.625	5.775
6	4	10	0.5	5.325
6	4	10	1.5125	3.1875
6	4	10	1.1	6.1125
6	4	10	1.6625	5.6625

6	4	10	1.85	2.4
6	4	10.75	0.875	2.175
6	4	10.75	0.65	6.1125
6	4	10	0.9875	4.3125
6	4	10	1.625	1.5
6	4	10	0.95	2.7375
6	4	10	1.925	3.075
6	4	10	1.2875	6.1125
6	4	10.75	0.7625	3.6375
6	4	10.75	0.65	4.5375
11	6	26	1.61	3
13	8	30	0.95	4
21	11	23	1.18	5
15	4	24	1.97	6
9	11	35	1.99	5
18	10	19	1.36	4
14	14	34	1.85	3
16	7	18	1.91	3
14	14	30	1.24	2
10	7	16	0.74	3
16	14	37	0.6	3
19	8	25	1.94	2
23	12	22	1.18	3
19	6	38	0.82	2
12	11	32	1.29	6
23	4	21	1.29	6
15	5	16	1.48	2
21	4	25	1.58	4
15	5	15	1.2	2
6	8	31	0.78	5
23	6	39	1.62	3
20	5	31	1.27	3
16	12	17	0.61	5
17	14	38	1.88	6
11	4	24	1.42	3
10	14	22	1.08	3
8	11	22	1.36	6
18	8	32	0.82	3
20	6	26	1.59	3
14	4	39	0.89	2
10	9	31	1.28	3
10	6	39	1.65	2
15	13	36	1.52	3
6	11	16	1.62	3

7	6	29	1.07	2
10	9	24	1.37	2
6	12	16	0.55	2
14	7	37	1.27	2
13	6	10	1.25	3
10	6	12	1.02	2

Appendix E: Shortened Sequence Test Results: ATE evaluation per testing sequence

Configuration Name	KITTI Sequence									
	00m	01m	02m	03m	05m	06m	07m	08	09	10m
Gen1	0.8999	3.5947	4.7502	0.2572	0.4234	0.5789	0.4203	4.0381	4.7562	1.1644
Gen2	1.2519	4.0831	5.1117	0.2346	0.4974	0.8558	0.3716	3.9386	4.1466	0.5595
Gen3	1.0755	0.0000	3.8339	0.2495	0.3397	0.7361	0.4845	0.0000	1.5736	1.0084
Gen4	0.9779	4.5977	4.2080	0.2476	0.3831	0.7716	0.4596	3.0455	1.4884	1.1004
Gen5	0.9505	4.1742	4.4417	0.2453	0.3724	0.7893	0.4013	3.1199	1.4948	0.9529
Gen6	0.9802	6.0234	4.0457	0.0000	0.3853	0.7746	0.4386	3.1709	1.5114	1.0852
Gen7	0.8883	4.1988	4.5948	0.2581	0.4576	1.0311	0.3005	3.4581	3.9998	0.6311
Gen8	1.0198	4.5316	3.8826	0.2390	0.3653	0.7214	0.4691	2.9561	1.5013	0.9955
Gen9	1.0607	3.2427	4.4622	0.2505	0.3626	0.7238	0.4347	3.2461	1.5604	1.2179
Gen10	1.0203	3.3488	4.4106	0.2554	0.4270	0.9801	0.4316	4.2856	4.2812	0.7228
Gen11	0.9698	8.0076	3.9657	0.2331	0.3826	0.8136	0.4775	3.0807	1.4773	0.9845
Gen12	0.9666	3.4525	4.6159	0.2414	0.3685	1.4295	0.4446	3.0949	1.4935	0.9085
Gen13	1.0563	1.4469	3.9760	0.2665	0.0000	0.7607	0.4665	3.5285	3.7477	1.6062
Gen14	0.8906	3.4928	3.9385	0.2564	0.4237	0.6576	0.4338	3.6338	4.2466	0.6162
Gen15	0.9587	3.2292	4.2209	0.2466	0.3827	0.7268	0.4266	0.0000	1.4629	1.0505
Gen16	0.9474	4.2513	3.7547	0.2560	0.3685	0.6845	0.4590	2.9748	1.4661	1.0447
Gen17	1.1219	1.6016	4.3122	0.2333	0.5270	0.8073	0.4257	3.4473	4.0724	0.7087
Gen18	0.9856	4.8574	3.9162	0.2513	0.3577	1.3334	0.3976	3.1692	1.5633	1.1027
Gen19	0.9718	7.3836	4.1437	0.2585	0.4099	0.7141	0.4574	3.3300	1.3983	0.9864
Gen20	1.0907	4.7155	4.3216	0.0000	0.4105	0.8310	0.4955	3.1615	1.5091	1.2641
Gen21	0.9369	3.0545	4.5332	0.2427	0.3663	0.7134	0.4413	3.2061	1.5154	0.8510
Gen22	1.0025	4.3145	3.9041	0.2271	0.3784	0.7809	0.4500	3.0636	1.5976	1.0891
Gen23	0.9533	4.9531	3.8465	0.2486	0.3789	0.8182	0.4605	3.2414	1.5336	1.0946
Gen24	0.9910	4.2064	3.8321	0.2450	0.3937	0.7137	0.4467	3.2297	1.5672	1.0176
Gen25	0.9609	5.2168	4.5491	0.2424	0.3965	0.4437	0.5225	3.0934	1.4999	1.2143
Gen26	1.0164	4.8534	3.8296	0.2343	0.4121	0.7399	0.4710	3.0892	1.5770	1.1128
Gen27	0.9851	4.1362	3.9910	0.2268	0.3682	0.7767	0.4356	2.9514	1.4675	1.2266
Gen28	0.9571	4.1704	4.1540	0.2467	0.4116	0.7250	0.4321	2.9718	1.5374	0.9456
Gen29	0.9660	1.8560	3.8460	0.2473	0.4553	0.8216	0.4796	3.2561	3.7701	0.2690
Gen30	1.0294	4.5032	4.3058	0.2621	0.3853	0.9830	0.4484	3.2371	1.5418	0.8891
Gen31	1.1095	4.5302	5.0135	0.2356	0.4380	0.9162	0.4365	3.9102	4.3431	1.9484
Gen32	1.0978	3.4088	4.1840	0.2536	0.3700	0.7514	0.4563	3.2268	1.4923	1.0312
Gen33	0.9031	3.6246	3.6415	0.2471	0.4842	0.0000	0.4327	3.6825	4.3716	0.6712
Gen34	0.9155	0.0000	4.1851	0.2634	0.3960	0.7603	0.4590	2.8948	2.6731	1.0578
Gen35	1.0288	4.0374	4.0715	0.2525	0.3805	0.7446	0.4462	3.3381	1.5869	1.0381
Gen36	0.9010	5.2018	4.7587	0.2344	0.4609	0.9768	0.3641	3.4537	3.6917	0.7341
Gen37	0.9533	3.5940	4.0139	0.2434	0.4228	0.8648	0.4234	3.3605	1.4776	1.0069
Gen38	0.9861	4.0464	3.9584	0.2578	0.4144	0.7800	0.4813	3.2609	1.5372	1.0456
Gen39	1.0000	2.2627	4.2479	0.2450	0.3677	0.7010	0.4520	3.5687	3.7338	0.7722
Gen40	1.2481	5.3331	4.0322	0.2555	0.4356	0.7334	0.4727	3.2943	3.9108	1.8404
Gen41	0.9943	4.5386	4.4172	0.2406	0.3815	0.9576	0.4490	0.0000	1.5180	0.9772
Gen42	1.5646	3.4013	4.9450	0.2419	0.0000	0.8856	0.4638	3.7365	4.1467	0.9509
Gen43	0.9682	3.0979	0.0000	0.2501	0.4942	0.6600	0.3422	3.5180	4.7583	0.6824

Gen44	0.9703	0.0000	4.3019	0.2461	0.4103	0.7316	0.4585	2.8963	0.0000	1.0750
Gen45	1.0357	2.2727	4.1748	0.2485	0.4271	0.7612	0.4446	0.0000	3.2351	0.7595
Gen46	0.9636	4.2413	4.1739	0.0000	0.3810	0.8101	0.4399	3.3788	1.5095	1.2798
Gen47	1.0200	3.2230	4.0526	0.2549	0.4087	0.7567	0.4738	3.2846	3.5258	0.8534
Gen48	1.3528	1.3539	0.0000	0.2360	0.5615	0.8556	0.6046	3.3728	3.3322	0.6936
Gen49	0.9702	2.5124	4.1222	0.2531	0.0000	0.7199	0.4087	3.2677	1.5298	1.0070
Gen50	0.9607	4.7609	3.8485	0.0000	0.4003	0.7343	0.4796	3.1387	3.4928	0.7761
Gen51	0.9361	3.7379	4.0460	0.2436	0.3648	0.6668	0.4422	3.2165	1.5647	1.1493
Gen52	0.9341	1.7422	3.9773	0.2351	0.4336	0.8545	0.3997	3.7236	3.9345	1.3095
Gen53	1.2218	5.1906	4.0267	0.2550	0.3736	0.6138	0.4633	3.0582	3.6966	0.2861
Gen54	1.0092	4.2481	4.3980	0.2632	0.3888	0.7617	0.4599	3.0639	1.5166	0.9888
Gen55	1.1165	2.6262	0.0000	0.2323	0.4705	0.7028	0.3505	3.6914	4.0251	0.6621
Gen56	1.1771	2.1334	4.1429	0.2526	0.4052	0.7433	0.4671	3.2730	3.9230	0.8991
Gen57	1.0450	2.3820	3.5538	0.2552	0.4413	0.7235	0.4561	3.1764	3.3708	0.8299
Gen58	0.9361	4.1045	5.4133	0.2613	0.4907	0.8869	0.4457	3.7130	4.6624	1.1873
Gen59	1.3341	4.0666	4.0120	0.2382	0.4916	1.1684	0.3998	3.9786	4.5638	0.6238
Gen60	1.0007	6.5335	4.2735	0.2641	0.4002	0.8312	0.4700	3.3454	1.5718	1.1460
Gen61	1.2649	4.6248	5.2696	0.2484	0.4442	0.8255	0.4821	4.2092	4.6000	0.8213
Gen62	0.8912	1.5184	5.0916	0.2167	0.4677	0.8925	0.3590	3.5885	3.9740	0.3078
Gen63	1.3305	2.2626	4.9486	0.2262	0.4955	0.9833	0.3769	3.8077	4.4011	0.2938
Gen64	0.9468	2.6607	4.1757	0.2588	0.3821	0.6331	0.4266	3.1646	1.6207	1.1000
Gen65	1.0761	5.3101	4.2171	0.2474	0.3994	0.7092	0.4899	3.3375	1.4301	1.0396
Gen66	0.9219	1.1925	4.6890	0.2305	0.4601	0.6977	0.3548	4.1000	4.3204	0.8314
Gen67	1.0013	2.8535	5.5591	0.2249	0.0000	0.7875	0.4311	3.7436	3.7429	0.6291
Gen68	0.8913	3.3705	4.3080	0.2509	0.3496	0.6589	0.4448	3.1650	1.5606	1.1816
Gen69	1.1010	4.6303	3.7123	0.2514	0.4074	0.7262	0.4463	3.6024	3.8620	0.0000
Gen70	0.0000	5.6859	3.9306	0.2709	0.4156	0.7173	0.4618	3.3521	3.9906	0.8426
Gen71	1.0131	3.5586	4.2204	0.2384	0.3720	0.0000	0.4303	3.2795	1.5239	1.0080
Gen72	0.0000	2.6880	4.1644	0.2509	0.4057	0.7214	0.4619	3.1480	4.2114	1.3764
Gen73	0.9507	3.1270	4.1112	0.2098	0.4861	0.5400	0.3627	4.1505	4.9157	0.7027
Gen74	1.0173	2.3004	5.0450	0.2645	0.0000	0.6665	0.5019	3.8328	4.2246	0.5307
Gen75	0.9165	4.5734	3.9447	0.2567	0.3633	0.7383	0.4466	3.3288	3.1662	1.0727
Gen76	0.9556	4.1013	4.1798	0.2406	0.3689	0.7366	0.4305	3.4728	1.4947	0.8722
Gen77	0.9289	4.0797	3.9452	0.2632	0.3819	0.7476	0.4618	3.3882	3.6314	1.0061
Gen78	1.5788	7.4256	4.1269	0.2511	0.3740	0.7113	0.4341	2.8387	1.5449	1.2328
Gen79	1.0481	2.9664	4.1835	0.0000	0.4014	0.7571	0.4246	3.2467	1.5353	1.0737
Gen80	0.9338	1.9770	4.5736	0.2534	0.4767	0.8635	0.3890	3.8030	4.3527	0.5892
Gen81	0.9311	3.7031	4.0290	0.2389	0.4474	1.1502	0.3943	3.4443	4.5072	0.5859
Gen82	0.0000	6.8699	4.1677	0.0000	0.3736	0.7305	0.4739	3.2870	1.4636	1.1497
Gen83	0.9040	3.2886	0.0000	0.2439	0.5069	0.8451	0.5213	4.1568	4.2176	0.5685
Gen84	0.9086	2.9544	4.1468	0.2604	0.4250	0.9906	0.3377	3.8067	4.3172	0.6201
Gen85	1.3162	2.0389	4.9205	0.2465	0.4575	0.9362	0.3939	3.3754	4.7088	0.6530
Gen86	1.0282	3.4686	5.5376	0.2320	0.4783	0.7189	0.4033	4.4201	4.4431	1.0609
Gen87	1.0330	4.2841	4.1255	0.2649	0.3991	0.6947	0.0000	3.1659	3.5942	0.0000
Gen88	1.0124	5.0222	3.6473	0.2453	0.3927	0.7094	0.4666	3.2754	3.2816	0.2802

Gen89	1.0363	4.0620	4.0763	0.2388	0.3881	0.7992	0.4698	2.9589	1.5799	1.0150
Gen90	0.9569	3.5095	4.2437	0.2603	0.4110	0.7846	0.4801	3.2269	3.2302	0.2931
Gen91	0.9889	4.2742	3.9468	0.2486	0.3860	0.8414	0.4680	3.0225	1.5189	1.1666
Gen92	1.3133	3.4756	5.6966	0.2682	0.4063	0.9403	0.3709	4.1162	3.6178	0.7129
Gen93	0.9901	2.8498	3.7072	0.2560	0.3959	0.7251	0.4528	3.0172	3.5128	0.9093
Gen94	1.3865	3.0936	4.7071	0.2418	0.4935	0.8192	0.4216	4.0612	4.7520	0.0000
Gen95	1.0206	4.0242	4.4162	0.2342	0.4160	0.8130	0.4338	3.2755	1.4836	0.9436
Gen96	0.9549	2.0507	4.2202	0.2458	0.3836	0.7234	0.4622	0.0000	1.5606	1.1501
Gen97	0.9714	3.5947	4.0488	0.2672	0.3812	0.7047	0.4692	0.0000	3.8808	0.2768
Gen98	0.9265	4.5597	4.0656	0.2446	0.0000	0.6908	0.4090	3.1295	3.9165	0.8314
Gen99	1.0640	2.1306	3.7964	0.2392	0.3828	0.7454	0.4652	3.2043	3.7966	0.5651
Gen100	0.9385	4.8142	4.3828	0.2607	0.3958	0.6532	0.4209	3.1526	1.4780	1.0621
Gen101	0.9768	3.6725	3.9136	0.2513	0.3918	0.7566	0.4902	3.1867	4.0014	0.9400
Gen102	0.9887	4.6488	3.8551	0.2406	0.0000	0.7642	0.4499	3.0162	1.4849	1.1144
Gen103	0.9977	2.5026	3.9177	0.2654	0.3768	0.7333	0.4974	3.0311	1.6245	0.9288
Gen104	1.0443	7.3668	4.1219	0.2392	0.3847	0.8496	0.4281	3.1464	1.5205	1.0204
Gen105	0.9340	2.8499	3.9319	0.2321	0.4245	0.7299	0.4615	2.9590	3.7102	0.8901
Gen106	0.9094	3.3380	3.8540	0.2476	0.0000	0.7539	0.4458	3.2086	3.7564	0.7350
Gen107	1.0634	4.2725	3.4966	0.2589	0.4202	0.6982	0.4775	3.1761	4.0676	0.2929
Gen108	1.1440	2.5415	4.7788	0.2331	0.4789	0.9393	0.4503	4.0168	4.3328	0.6456
Gen109	1.1685	3.4255	5.1775	0.2736	0.5164	0.6371	0.0000	3.8711	4.9920	0.3467
Gen110	0.9896	2.5729	4.9024	0.2295	0.4668	0.9608	0.3811	3.6692	4.3714	0.6506
Gen111	0.9889	2.6227	3.6380	0.2463	0.3921	0.6679	0.4667	3.4367	3.5843	0.2950
Gen112	0.9741	3.3063	4.2391	0.2352	0.3598	0.7761	0.4722	3.2162	1.6132	1.1183
Gen113	0.9981	3.0376	4.5754	0.2543	0.3990	1.0126	0.4991	3.9105	3.8961	0.3277
Gen114	1.0165	4.4076	4.2226	0.2554	0.3698	0.8011	0.4386	0.0000	1.5951	1.0691
Gen115	2.0785	5.1316	4.0795	0.2542	0.4128	0.7340	0.0000	3.3585	1.4784	1.0497
Gen116	0.9103	6.2186	3.8195	0.2505	0.3706	0.7394	0.0000	3.0318	1.4849	1.1958
Gen117	1.1729	2.3484	4.5293	0.2356	0.4580	0.8385	0.3576	4.1574	4.5598	0.6027
Gen118	0.8927	0.0000	4.0961	0.2644	0.4874	0.7022	0.4702	4.0060	4.3891	0.6154
Gen119	0.0000	0.0000	4.7930	0.2456	0.4488	0.6938	0.3425	4.1117	4.4647	0.8704
Gen120	0.9418	6.5712	3.8690	0.2327	0.3710	0.8252	0.4566	3.1844	1.5178	1.1038
Gen121	1.0831	2.0488	4.3990	0.2326	0.3966	0.8278	0.4769	3.2299	0.0000	0.2908
Gen122	1.1154	2.5230	4.0517	0.2482	0.4079	1.9424	0.4127	3.2649	1.5726	1.2117
Gen123	0.9278	3.8246	4.3994	0.2281	0.3772	0.7807	0.4271	3.1213	1.5272	1.0834
Gen124	1.2883	5.3519	4.1158	0.2558	0.3728	0.7580	0.4918	2.8981	1.5045	1.2775
Gen125	1.0078	5.7212	4.0900	0.2366	0.3724	0.7716	0.4243	3.1486	2.9027	1.1612
Gen126	1.0113	3.1975	4.1091	0.2673	0.3987	0.6987	0.4288	3.0067	1.3999	1.0666
Gen127	1.1659	1.6794	4.1848	0.2527	0.0000	0.7515	0.5150	3.3700	3.8774	0.8333
Gen128	0.9072	3.7631	4.1948	0.2529	0.4358	0.7378	0.4478	3.4554	3.7353	0.2722
Gen129	1.1654	2.2862	0.0000	0.2222	0.4442	1.1714	0.4432	4.4188	4.1925	0.7083
Gen130	1.0788	2.0737	3.7244	0.2386	0.4049	0.7142	0.4857	0.0000	3.6613	0.7726
Gen131	1.0269	4.3483	3.9740	0.2505	0.3756	0.9705	0.4679	3.0817	1.5235	1.0025
Gen132	0.9380	4.6714	4.2441	0.2476	0.4631	0.7924	0.4560	3.7626	4.1469	0.5989
Gen133	0.9695	5.8385	4.0578	0.2480	0.3767	0.7725	0.4636	3.3102	1.5915	0.9943

Gen134	1.0532	2.7136	4.2411	0.2503	0.3715	1.0749	0.4278	3.2168	1.5773	1.1717
Gen135	1.0638	3.7096	4.0883	0.2350	0.4174	0.6817	0.3429	3.5025	4.0785	0.6156
Gen136	1.2556	3.1614	4.0725	0.2523	0.3912	0.7257	0.4394	2.9507	1.4648	1.3503
Gen137	1.1727	3.4103	4.7768	0.2300	0.0000	1.1311	0.4080	0.0000	4.0122	0.6313
Gen138	0.0000	1.7633	4.0775	0.2678	0.3931	0.7206	0.4896	3.4229	3.4382	0.3068
Gen139	1.3851	1.3853	5.5078	0.2291	0.5705	1.0141	0.6178	3.2862	4.4933	0.7431
Gen140	0.9191	4.7315	4.1623	0.2547	0.0000	0.5944	0.4362	3.5204	3.3072	0.9581
Gen141	0.8968	1.4401	4.2331	0.2530	0.4435	0.6214	0.4395	3.1793	4.1384	0.2889
Gen142	1.0514	1.3372	4.3422	0.2275	0.4344	0.7337	0.4754	3.1754	3.7994	0.5817
Gen143	0.9116	2.9455	4.1021	0.2331	0.4558	0.6089	0.4227	3.4011	3.8091	0.8246
Gen144	0.9780	3.7692	4.3791	0.2389	0.3882	0.7033	0.4344	3.3447	1.5328	1.2035
Gen145	0.9673	2.7683	3.8991	0.2409	0.3923	0.6441	0.4135	3.0325	1.4255	1.2176
Gen146	1.1198	4.1717	4.2671	0.2363	0.4104	0.7650	0.4673	3.2261	1.5102	0.3118
Gen147	1.0002	4.2901	3.9987	0.2400	0.3809	0.7163	0.4562	3.2454	1.4941	1.1538
Gen148	0.8771	3.9477	4.8579	0.2403	0.4374	1.0202	0.4379	4.0912	0.0000	0.7702
Gen149	1.8659	1.9935	4.1804	0.2565	0.4071	0.6814	0.4584	3.1963	3.9037	0.2920
Gen150	0.9575	3.5574	4.2911	0.2386	0.4135	0.8942	0.4233	3.0934	1.5912	1.0363
Gen151	0.9889	0.0000	4.3118	0.2515	0.3791	0.7627	0.4316	2.9993	1.6375	1.1561
Gen152	1.0477	3.5487	3.7168	0.2301	0.3755	0.9786	0.4560	3.2742	1.5297	0.9360
Gen153	0.9768	5.2458	4.3975	0.2581	0.3894	0.7148	0.4118	3.3314	1.6138	1.1381
Gen154	1.0137	2.4929	4.0943	0.2196	0.3492	0.7416	0.4047	3.1601	1.5482	1.1145
Gen155	0.9003	4.3482	4.0851	0.2460	0.4013	0.7483	0.4891	2.7282	1.4731	1.1268
Gen156	0.9532	0.0000	4.2343	0.2284	0.3747	0.7673	0.4310	3.2153	3.6587	0.0000
Gen157	1.1580	1.6535	0.0000	0.2547	0.4519	0.9146	0.3947	4.2185	4.4228	0.8554
Gen158	0.9563	0.0000	4.3413	0.2570	0.3902	0.7336	0.4300	3.1708	1.6468	1.0348
Gen159	1.0370	3.3692	4.0540	0.2442	0.4350	0.7592	0.4956	3.6344	0.0000	0.5602
Gen160	0.8661	3.6479	4.5947	0.2297	0.3961	0.7199	0.4495	3.0366	1.4613	1.1962
Gen161	0.0000	2.3401	4.0240	0.0000	0.3899	0.6966	0.4597	3.2503	1.6423	0.7915
Gen162	0.9673	5.5556	4.0848	0.2451	0.4050	0.9346	0.4617	2.8543	1.5129	0.9842
Gen163	0.9446	1.9656	3.9026	0.0000	0.4514	0.7417	0.4923	3.3686	3.6057	0.6697
Gen164	1.0391	4.9746	4.0665	0.2562	0.4405	0.7371	0.4533	3.2359	3.8855	0.7149
Gen165	0.9681	3.5366	4.3467	0.2336	0.3715	0.7519	0.4625	3.4804	1.5479	1.0740
Gen166	0.9408	3.0139	4.2160	0.2439	0.3967	0.7335	0.4800	2.9278	1.5580	1.0839
Gen167	1.0135	1.7272	4.6346	0.2428	0.5460	1.0388	0.6838	3.5741	4.4828	0.7855
Gen168	1.1744	5.2755	4.2919	0.3262	0.5948	0.5926	0.4577	3.6477	3.2636	1.5496
Gen169	0.9483	2.4945	3.7426	0.2418	0.4125	0.8838	0.4385	3.1845	3.8940	0.7775
Gen170	1.3958	2.9582	3.7617	0.2430	0.4078	0.6850	0.4504	3.1312	0.0000	0.8060
Gen171	0.9600	3.3894	3.5634	0.2557	0.3709	0.7512	0.0000	3.3168	3.6747	0.5767
Gen172	1.2932	4.4752	4.2035	0.2879	0.3613	0.5895	0.4407	3.3189	3.8132	1.2241
Gen173	0.9629	2.8206	5.0237	0.2506	0.4162	0.5361	0.4636	3.6869	3.0380	1.2579
Gen174	1.0441	3.8477	4.3117	0.2487	0.3919	0.6710	0.4500	3.0432	1.5296	1.0848
Gen175	0.9953	7.3804	3.9545	0.2418	0.3677	0.8276	0.4752	3.1556	1.5102	1.1569
Gen176	0.0000	2.3969	4.4861	0.2502	0.5279	1.0183	0.5846	3.6598	4.9269	0.2988
Gen177	0.9261	4.6818	3.8218	0.2544	0.3642	0.8486	0.5057	3.1559	0.0000	1.1573
Gen178	1.1794	1.4356	3.5525	0.2381	0.3940	0.8591	0.4642	3.3041	3.5162	0.6496

Gen179	0.9806	6.1883	3.8745	0.2521	0.3895	0.7646	0.4712	3.1075	3.3206	1.0164
Gen180	1.0107	4.2396	4.0896	0.2601	0.3561	0.7096	0.4525	3.0613	3.9319	0.7754
Gen181	1.0075	2.5443	4.0785	0.2481	0.3780	0.7092	0.4478	3.5404	1.5356	1.1460
Gen182	0.9766	5.3014	3.6239	0.2369	0.4044	0.8064	0.4298	2.9694	1.4522	1.0152
Gen183	0.8598	3.3221	4.3206	0.2580	0.3987	0.8881	0.4375	3.1884	1.6435	1.1405
Gen184	1.0491	2.8725	5.8213	0.2805	0.4178	0.0000	0.5898	4.1977	4.2490	1.3822
Gen185	1.1391	4.5132	3.6882	0.2399	0.4048	0.8534	0.4795	3.0340	1.5994	1.0794
Grd1	0.9648	1.9527	4.3619	0.2634	0.5534	1.1288	0.5893	3.2891	4.3231	0.5660
Grd2	0.9778	2.8030	4.0604	0.2595	0.4988	1.1057	0.5095	3.6369	4.4790	0.6549
Grd3	1.3736	4.2460	3.9314	0.2234	0.4513	0.6074	0.4122	2.9713	4.0153	0.6151
Grd4	1.0416	2.9815	3.3189	0.2501	0.0000	0.0000	0.4132	3.2528	3.7410	0.7910
Grd5	0.7974	1.3477	0.0000	0.2664	0.5086	0.0000	0.5302	3.7349	4.4666	0.5510
Grd6	0.9479	1.3433	3.6272	0.2971	0.4771	1.0601	0.5537	3.8821	4.2742	0.7716
Grd7	0.9329	1.5339	4.0803	0.2458	0.4412	1.1412	0.5074	3.4498	4.5824	0.7654
Grd8	0.9270	1.3336	3.5516	0.2458	0.4656	1.1083	0.5422	3.0054	3.9680	0.6634
Grd9	1.0075	1.3210	3.8030	0.2519	0.5909	0.8900	0.5032	3.4432	4.7294	0.6472
Grd10	1.0100	1.3576	5.2567	0.2742	0.4992	1.0482	0.5224	3.6755	3.5734	0.6775
Grd11	0.9076	2.8716	3.5737	0.2727	0.4994	0.8325	0.5163	3.5413	5.0321	0.6160
Grd12	1.3403	1.2842	3.3706	0.2310	0.4038	0.6163	0.4273	3.2081	3.9440	0.7462
Grd13	1.0452	2.0791	3.8914	0.2698	0.5581	1.1309	0.5170	3.6216	4.0409	0.7249
Grd14	1.5144	1.3436	3.5279	0.2539	0.5130	0.9650	0.5405	3.2545	3.8019	0.6938
Grd15	0.9562	1.9545	4.0856	0.3046	0.4963	0.7947	0.5041	4.4104	4.5556	0.5556
Grd16	1.1997	3.0078	3.6024	0.2489	0.4077	0.8417	0.4170	3.0190	3.2948	0.8073
Grd17	0.9427	2.2157	3.5969	0.2573	0.5059	1.0378	0.5401	3.3949	5.2751	0.7540
Grd18	0.9816	2.7093	0.0000	0.2642	0.3992	0.7229	0.4450	3.4240	3.6948	0.2857
Grd19	1.3455	3.8179	4.6147	0.2470	0.5087	0.7204	0.5204	3.9760	5.1755	0.6525
Grd20	0.9614	3.4070	3.4916	0.2651	0.4647	0.6616	0.6513	3.6980	4.4844	0.5607
Grd21	0.9284	1.1744	4.5524	0.2672	0.5358	0.9865	0.4816	3.8157	3.8585	0.7227
Grd22	1.0784	3.0608	4.7073	0.2768	0.5169	0.9811	0.5784	3.4415	4.8811	0.7572
Grd23	1.4069	1.3192	3.5650	0.2510	0.4039	0.8803	0.4205	3.2036	3.4717	0.9861
Grd24	0.8773	1.4366	3.6564	0.2523	0.4676	1.1179	0.4844	3.6704	4.6520	0.6743
Grd25	0.8972	1.1785	3.9645	0.2607	0.4467	0.6159	0.4329	3.1264	3.9649	0.6366
Grd26	0.9370	4.7869	4.0438	0.2572	0.5360	0.9829	0.5751	3.7407	4.0339	0.6055
Grd27	0.9416	1.4766	3.5341	0.2507	0.4564	1.4718	0.4461	3.3428	3.6562	1.4943
Grd28	0.9674	3.8037	4.4970	0.2652	0.5375	0.7975	0.4639	3.4315	4.0513	0.3021
Grd29	0.9142	1.9191	4.1860	0.2512	0.4406	0.9027	0.0000	3.2307	4.5550	0.6661
Grd30	1.0524	1.1369	4.1615	0.3015	0.5471	0.9120	0.5402	3.3475	4.0113	0.6472
Grd31	0.0000	4.5195	4.2720	0.2522	0.5256	0.9255	0.5558	3.3334	4.5830	0.6293
Grd32	0.9557	2.4688	4.0049	0.2533	0.4072	0.6007	0.4940	3.2242	3.8259	0.6434
Grd33	1.2695	2.3402	0.0000	0.2465	0.4894	0.9939	0.5004	3.3823	4.5268	0.6864
Grd34	0.9806	3.3007	3.5692	0.2805	0.5550	0.9010	0.4350	3.6688	4.6288	0.2820
Grd35	0.9671	1.5367	3.6347	0.2371	0.4013	0.7714	0.4388	3.3474	3.4363	0.2969
Grd36	0.9473	1.4529	3.5559	0.2441	0.4039	0.7630	0.4661	3.1419	3.6534	1.2040
Grd37	1.1980	1.2742	3.8071	0.2667	0.4938	1.0233	0.5724	3.2522	4.7742	0.5890
Grd38	1.0003	1.3278	3.1170	0.2544	0.4625	0.5574	0.4692	3.1775	3.9646	0.6945

Grd39	0.9481	1.4783	5.0270	0.2482	0.4931	0.8480	0.6096	3.7372	4.1654	0.6206
Grd40	0.9389	1.1597	3.2392	0.2477	0.4424	0.5425	0.4713	3.0793	3.8067	0.6733
Grd41	1.0263	3.4165	3.9112	0.2603	0.5639	1.0676	0.4696	3.2643	4.8147	0.7019
Grd42	0.9582	1.8296	4.5733	0.2624	0.5412	0.9550	0.6370	3.6792	4.5439	0.2989
Grd43	0.9451	2.0922	3.9259	0.2458	0.4184	0.7448	0.4822	3.1480	3.7623	1.4517
Grd44	1.0648	1.2787	3.9030	0.2640	0.5122	0.8111	0.5560	3.0209	4.4315	0.6172
Grd45	0.9182	1.5789	3.7836	0.2513	0.4424	0.7198	0.4578	3.1444	3.7000	0.9409
Grd46	0.9453	1.2666	3.7784	0.2539	0.4202	0.8925	0.5438	3.4083	4.2738	0.6644
Grd47	1.1080	2.3060	3.8123	0.2472	0.4008	0.8371	0.4542	3.1262	3.6537	0.8211
Grd48	0.9453	1.6194	4.0232	0.2528	0.4481	0.6521	0.4578	3.2390	3.8511	0.6919
Grd49	1.0185	1.3157	4.2369	0.2631	0.5160	0.9498	0.4579	3.3764	0.0000	0.6618
Grd50	1.4008	1.5362	3.6278	0.2565	0.3851	0.7990	0.0000	3.1630	3.7733	0.8303
Grd51	1.3267	1.5057	3.6136	0.2622	0.3915	1.4273	0.4313	3.1058	0.0000	0.7794
Grd52	0.9649	3.3119	3.5642	0.2525	0.5257	0.8909	0.4746	3.2970	0.0000	0.6777
Grd53	1.0624	2.2887	4.4808	0.2881	0.5623	1.0160	0.5302	3.4923	3.8512	0.6853
Grd54	1.0816	1.5149	4.0343	0.2636	0.5187	1.0127	0.0000	3.7184	5.2080	0.7092
Grd55	1.4462	1.8171	3.8087	0.2594	0.4315	0.7560	0.4469	3.4125	3.6270	0.7091
Grd56	1.3505	2.2683	5.1007	0.2533	0.4660	0.9312	0.4815	3.3548	4.4414	0.6425
Grd57	0.9640	2.4808	3.1503	0.2627	0.5640	1.2397	0.5302	3.1303	4.2128	0.7330
Grd58	1.1900	2.1634	4.1536	0.2402	0.5361	0.9153	0.4633	3.5375	4.5571	0.7673
Grd59	1.4169	1.2739	3.7452	0.2450	0.4099	0.7382	0.4427	3.2275	4.0452	1.0740
Grd60	0.9288	2.3910	3.6417	0.2648	0.5447	1.0943	0.6085	3.4491	4.6391	0.5792
Grd61	0.9724	4.9992	3.7377	0.2546	0.4305	0.7600	0.4428	3.1627	3.7034	0.8194
Grd62	1.2363	1.3766	4.6682	0.2928	0.5620	0.8308	0.4251	3.6933	4.4843	0.6515
Grd63	0.9266	1.3048	5.2183	0.2834	0.4756	1.1079	0.5654	3.6087	4.1877	0.5503
Grd64	1.3724	1.7759	3.9089	0.2493	0.4097	0.7509	0.4427	3.2305	3.8355	0.7893
Grd65	1.1093	1.5651	3.6933	0.2575	0.3966	0.7325	0.4396	3.4129	3.4382	0.7669
Grd66	1.1953	2.6275	4.2015	0.2288	0.4813	0.8558	0.5128	3.5232	3.6241	0.7309
Grd67	1.3867	2.2877	3.3884	0.2728	0.0000	1.0461	0.4083	3.6154	4.4842	0.6998
Grd68	0.9837	1.8435	4.0505	0.2515	0.4925	1.4220	0.5438	3.3426	4.4625	0.6194
Grd69	0.9886	3.2682	3.4630	0.2485	0.3911	0.9433	0.4246	3.3235	4.0095	0.6934
Grd70	0.9358	1.7901	4.3738	0.2363	0.4924	0.8524	0.4643	3.9334	3.8337	0.6941
Grd71	0.9558	1.2097	4.2614	0.2394	0.4350	0.5147	0.4606	3.1175	3.8145	0.6597
Grd72	1.1386	1.5226	4.5168	0.2691	0.5160	1.1106	0.6096	3.5371	4.6164	0.6356
Grd73	0.9824	1.4648	3.9241	0.0000	0.4354	0.7396	0.4477	3.1519	3.7323	1.2065
Grd74	1.0269	2.8861	3.4119	0.2340	0.4641	0.8498	0.4674	3.2469	3.9350	0.6318
Grd75	1.2818	2.8714	3.8973	0.2495	0.4258	0.6339	0.4166	2.9771	3.8363	0.6636
Grd76	1.1589	1.4275	4.5997	0.2346	0.5892	0.9479	0.4825	0.0000	4.3190	0.6799
Grd77	0.9404	1.2890	3.8718	0.2459	0.4153	0.7816	0.4702	3.3168	3.3890	0.7882
Grd78	1.6670	4.1445	3.9309	0.2692	0.5182	1.0525	0.5433	3.6560	4.7498	0.7229
Grd79	1.3704	2.0833	3.7782	0.2514	0.4128	0.7023	0.4809	3.1283	3.6215	0.8755
Grd80	0.9330	1.8646	3.6563	0.2501	0.3968	0.7539	0.4463	3.1282	1.5906	0.7755
Grd81	0.9282	1.5545	3.8296	0.2559	0.4401	1.0205	0.4763	3.3330	4.2151	0.7059
Grd82	0.9441	1.1474	4.7840	0.2687	0.4923	0.0000	0.5420	3.3703	4.0892	0.5746
Grd83	1.0326	1.8204	3.1664	0.2337	0.4780	0.6528	0.4170	3.3170	3.8659	0.5851

Grd84	1.0117	1.8870	3.3174	0.2565	0.5117	1.1450	0.5266	3.2757	4.5346	0.6162
Grd85	0.9054	0.0000	3.3548	0.2359	0.4354	0.5929	0.4789	3.1862	3.7138	0.7379
Grd86	1.0966	1.3213	3.6266	0.2290	0.4543	0.5918	0.4370	3.2508	4.0000	1.4710
Grd87	1.2094	1.1344	3.3556	0.2485	0.3971	0.7329	0.4560	3.2687	3.6708	0.2798
Grd88	0.9224	2.3539	4.9810	0.2574	0.5300	1.1352	0.5913	3.5796	4.3759	0.3180
Grd89	1.0460	1.2667	3.5987	0.2290	0.4109	0.7929	0.4569	3.1962	3.7948	1.8706
Grd90	1.3880	1.2476	4.3825	0.2609	0.5160	1.0274	0.4903	3.2776	4.4268	0.6593
Grd91	1.0278	1.8894	3.3474	0.2795	0.5654	1.0833	0.5498	3.8057	4.3096	0.6978
Grd92	0.9726	1.7472	3.6044	0.2440	0.3793	0.7175	0.4786	3.2309	3.6445	0.7479
Grd93	1.0170	1.2042	3.8391	0.2456	0.4328	0.7719	0.3546	3.2093	3.4978	0.2816
Grd94	0.9123	1.2999	3.6485	0.2423	0.4250	0.6132	0.4563	3.3647	4.1663	0.7407
Grd95	0.9134	1.6766	3.6456	0.2477	0.3912	1.6708	0.4670	3.1287	3.7110	0.7981
Grd96	0.9492	2.8462	4.1426	0.2412	0.4769	0.9912	0.5145	3.4601	4.5323	0.3209
Grd97	0.9388	1.8571	3.4155	0.2629	0.5007	0.9156	0.5628	3.6120	4.7372	0.6291
Grd98	0.9322	1.6919	4.0290	0.2695	0.5475	0.9935	0.6257	3.2321	4.6041	0.6054
Grd99	1.4406	4.6660	3.9136	0.2443	0.5562	1.0743	0.4572	3.6107	4.7995	0.6055
Grd100	0.9615	2.2190	4.3036	0.2531	0.4734	0.8721	0.4890	0.0000	4.1440	0.5948
Grd101	1.2338	1.2857	4.0083	0.2381	0.4633	0.5904	0.4290	3.1355	3.8791	0.2950
Grd102	1.1626	2.5046	3.6194	0.2730	0.5358	1.0053	0.5586	3.6940	4.5048	0.6873
Grd103	1.0916	1.3226	3.9671	0.2508	0.5650	0.7626	0.5749	3.5674	4.3864	0.7437
Grd104	0.9305	1.6896	3.7531	0.2314	0.4642	0.7860	0.4433	3.1452	3.8870	0.6708
Grd105	1.0938	0.0000	3.6679	0.2383	0.4000	0.6012	0.4251	3.0137	3.6790	0.0000
Grd106	1.2410	1.8710	3.9506	0.2591	0.4662	1.1952	0.4027	3.1175	3.6027	0.7556
Grd107	0.0000	1.8158	4.0273	0.2511	0.0000	0.8842	0.5301	3.5252	4.2950	0.8043
Grd108	1.5325	1.3680	0.0000	0.2427	0.3853	0.7904	0.4306	3.1215	3.6647	2.6634
Grd109	1.4577	3.8977	4.6541	0.2739	0.4886	1.1955	0.4781	3.5749	4.0641	0.6995
Grd110	0.9276	1.7076	3.6558	0.2433	0.4221	0.7173	0.0000	3.4668	4.3130	0.5880
Grd111	0.9569	2.7668	3.4813	0.2215	0.4861	0.6378	0.4340	0.0000	3.5878	0.6546
Grd112	1.2885	1.4080	3.4483	0.2637	0.5265	1.0460	0.5727	3.3235	4.4451	0.7315
Grd113	1.0326	1.6132	3.2985	0.2621	0.4019	0.7568	0.4451	3.3277	3.5355	0.7611
Grd114	1.0208	1.6662	3.7580	0.2457	0.4277	0.6183	0.3906	3.1612	3.6765	1.4532
Grd115	0.9291	3.9572	3.7400	0.2900	0.5197	0.9517	0.5651	3.5151	4.8781	0.6464
Grd116	0.9631	1.8807	3.4792	0.2489	0.3891	0.8507	0.4334	3.4354	3.8348	0.7657
Grd117	0.8828	5.4071	3.6010	0.2545	0.4078	0.7543	0.4230	3.3963	3.8579	0.2881
Grd118	0.8739	2.2408	3.5588	0.2396	0.5266	0.7472	0.5819	3.3090	4.5507	0.6689
Grd119	0.8810	1.9339	5.1215	0.2427	0.5355	0.8488	0.5366	3.5002	4.2387	0.6086
Grd120	1.2218	2.1131	3.5075	0.2368	0.4072	1.7417	0.4656	3.0138	3.3141	0.7797
Grd121	0.0000	1.2675	3.3327	0.2479	0.3681	0.8183	0.4637	3.4874	3.3378	0.8667
Grd122	0.9502	2.2956	4.0957	0.2431	0.4573	0.8233	0.5902	3.1622	3.6065	0.5411
Grd123	0.9645	2.4810	3.6324	0.2519	0.4035	0.8391	0.4344	3.4147	3.4314	1.4818
Grd124	0.9706	1.3403	4.0198	0.2458	0.4204	0.6061	0.4656	3.2943	3.4341	0.7977
Grd125	0.9759	2.5703	4.0417	0.2485	0.4117	0.7381	0.4493	3.2289	3.3617	2.0602
Grd126	1.4098	1.4381	3.5652	0.2486	0.3705	0.8215	0.4275	3.2907	3.5538	0.7889
Grd127	0.8743	1.3772	3.5621	0.2353	0.4166	0.6859	0.4373	3.0382	1.6096	0.0000
Grd128	1.2466	1.9843	3.5883	0.2383	0.3834	0.6379	0.4129	3.0207	3.9657	0.0000

Grd129	0.0000	1.4883	3.8011	0.2488	0.4617	0.6723	0.4734	2.9641	3.7374	0.0000
Grd130	0.8379	1.3197	3.5630	0.2516	0.3840	1.6927	0.4301	3.2173	3.3098	0.7157
Grd131	0.8804	1.9504	3.4048	0.2555	0.3993	0.7693	0.4446	3.4570	3.6174	0.2770
Grd132	0.9650	2.0383	3.5197	0.2519	0.4664	0.6140	0.4438	3.1666	4.0078	0.7914
Grd133	1.3715	1.9765	3.3937	0.2274	0.4182	0.7975	0.4625	3.2629	3.9408	0.2884
Grd134	1.0214	2.9634	3.8439	0.2516	0.4561	0.6571	0.4617	2.9222	3.5754	0.6561
Grd135	0.8844	2.1587	3.7978	0.2867	0.5159	0.8405	0.6406	3.1755	4.3216	0.6192
Grd136	0.9616	1.5849	3.7177	0.0000	0.4940	1.0421	0.5702	3.5059	4.9706	0.8099
Grd137	1.0182	3.8662	3.8398	0.2358	0.4444	0.5674	0.4978	3.4366	3.7726	0.7082
Grd138	0.9173	2.5200	4.2576	0.2341	0.4456	0.7167	0.4645	3.1987	3.7645	0.8081
Grd139	1.3316	1.5329	3.7678	0.2566	0.0000	0.7324	0.4765	3.0424	3.6712	0.2916
Grd140	0.9934	2.9393	3.8191	0.2562	0.3928	2.2644	0.4367	3.1493	3.5912	0.7996
Grd141	1.1123	2.1988	3.5256	0.2488	0.3889	0.8661	0.4801	3.4123	3.4210	0.8738
Grd142	1.0500	2.2217	3.9277	0.2591	0.4392	0.5616	0.4283	3.4017	4.0567	0.7792
Grd143	0.9685	1.2371	4.0985	0.2395	0.4498	0.5699	0.4444	3.0126	3.6486	0.2803
Grd144	1.0530	2.8573	3.8674	0.2670	0.4342	0.7241	0.4472	3.1410	3.3143	1.0003
Grd145	1.4720	2.3342	3.3555	0.2455	0.0000	0.7543	0.4796	3.2625	3.5767	0.0000
Grd146	1.3012	1.4338	3.4335	0.2234	0.4561	0.5935	0.4665	3.1993	1.4895	0.6453
Grd147	0.9075	1.8319	3.6118	0.2511	0.4224	0.7771	0.4312	3.2465	3.8108	1.0114
Grd148	1.1755	1.7878	0.0000	0.2563	0.4816	0.9442	0.0000	3.4088	0.0000	0.6536
Grd149	0.9719	1.4464	3.5763	0.2467	0.4316	0.5518	0.4593	3.2779	3.5920	0.6346
Grd150	0.9045	2.5359	3.3696	0.2192	0.4634	0.5709	0.4064	3.0176	3.8318	0.9590
Grd151	1.1158	1.5557	3.5426	0.2514	0.3961	0.7933	0.4366	3.2834	3.4993	0.8835
Grd152	0.9766	1.8563	3.8468	0.2544	0.4072	0.8145	0.4606	3.1715	3.7661	1.6936
Grd153	1.0752	1.5546	4.0339	0.2680	0.4360	0.9166	0.4537	3.2799	3.9268	0.6814
Grd154	1.3079	1.2877	4.5394	0.2284	0.0000	0.8076	0.5085	3.3698	4.8111	0.6814
Grd155	0.9725	1.2460	3.5186	0.2729	0.3912	0.6912	0.4478	0.0000	3.7275	0.7988
Grd156	1.0595	1.6541	4.5591	0.2689	0.5105	1.0953	0.4943	3.5460	3.9082	0.6575
Grd157	1.2543	2.3259	3.7119	0.2314	0.4643	0.9722	0.4333	3.1563	3.6739	0.6428
Grd158	1.3520	1.6421	3.6188	0.2367	0.4228	0.8365	0.4179	3.2623	3.2345	0.2880
Grd159	1.3426	1.1816	3.4727	0.2508	0.4397	0.6165	0.4547	3.0995	3.6954	0.6817
Grd160	1.1054	2.5113	4.1687	0.2389	0.4588	0.5816	0.4477	3.2553	3.7936	1.7094
Grd161	1.2392	1.4684	3.5237	0.2799	0.4530	1.3003	0.5893	3.6014	4.2818	0.8048
Grd162	0.9138	2.0306	3.8209	0.2602	0.4852	1.1037	0.5449	3.5813	4.8433	0.5784
Grd163	0.0000	1.5806	3.5070	0.2212	0.4613	0.8303	0.4658	3.2026	3.5755	0.6600
Grd164	0.9240	1.3056	3.0968	0.2612	0.5042	0.9420	0.5730	3.4368	4.3961	0.6899
Grd165	1.1987	2.5434	3.6911	0.2337	0.4131	0.6109	0.4555	3.3034	3.6335	0.7491
Grd166	1.0606	1.3878	3.5702	0.2546	0.4010	0.5967	0.4134	0.0000	4.2694	0.7119
Grd167	0.9467	1.3330	3.3336	0.2467	0.0000	0.8407	0.4499	3.3943	3.5131	0.2918
Grd168	1.0395	1.9737	3.8819	0.2607	0.4330	0.5309	0.4305	3.0184	3.8513	0.9792
Grd169	0.9064	1.3255	3.6432	0.2257	0.4545	0.6843	0.4455	3.0875	3.8649	0.5710
Grd170	0.9984	1.4981	4.1431	0.2681	0.4668	0.8573	0.4962	3.1312	4.2102	0.7854
Grd171	0.9426	1.6345	3.5730	0.2489	0.0000	0.7893	0.4517	3.2675	3.5768	0.2864
Grd172	0.9997	1.7403	3.3653	0.2479	0.4558	0.6304	0.4265	2.9900	4.0816	0.6994
Grd173	1.1249	1.8809	3.4724	0.2582	0.0000	0.6861	0.4410	3.4576	4.3432	0.6376

Grd174	0.9952	2.9240	3.8687	0.2607	0.4031	0.7841	0.0000	3.3464	3.8150	1.1947
Grd175	0.9373	1.6309	3.8475	0.2460	0.4228	0.7460	0.4948	3.2494	3.7877	0.8182
Grd176	1.0818	1.5333	3.7883	0.2440	0.4163	1.4108	0.4637	3.0093	4.0076	0.2722
Grd177	0.9577	2.3720	3.7841	0.2541	0.4076	1.0196	0.4671	3.2322	3.7181	0.5216
Grd178	0.9216	2.4984	3.7708	0.2516	0.4302	0.6942	0.4553	3.2573	4.0435	0.3132
Grd179	0.8918	1.5073	0.0000	0.2499	0.4457	0.6464	0.4368	3.0996	3.8668	1.3424
Grd180	0.9798	2.2096	3.8016	0.2491	0.4014	0.7409	0.4189	3.2669	3.6320	0.2916
Grd181	1.1057	1.5781	3.6758	0.2346	0.4137	0.7285	0.4195	3.1518	4.0305	0.6525
Grd182	1.1074	1.4035	3.7808	0.2465	0.0000	0.8487	0.4715	3.0290	0.0000	1.0580
Grd183	1.2869	2.1577	3.5015	0.2492	0.3666	0.9675	0.4937	3.1588	3.6168	0.7478
Grd184	1.0291	2.6145	3.5700	0.2365	0.3897	0.6992	0.4515	3.1775	3.5009	0.7295
Grd185	1.2797	5.3775	0.0000	0.2594	0.3961	2.1285	0.4752	2.9409	3.4815	0.2883
Grd186	0.9805	2.2508	3.4148	0.2532	0.3912	1.6559	0.4467	3.3985	3.7155	0.8714
Grd187	1.0332	1.4002	3.5806	0.2517	0.4016	0.7229	0.4624	3.1703	3.6933	0.7362
Grd188	1.0676	2.3795	3.8243	0.2556	0.3926	0.7259	0.4873	3.1451	3.5319	0.7904
Grd189	0.9517	1.4196	4.3436	0.2653	0.5181	0.9621	0.5155	3.3551	5.0591	0.5887
Grd190	1.0261	3.4513	3.6258	0.2564	0.4100	0.7343	0.4791	3.2661	3.6478	0.7454
Grd191	0.9309	1.3973	3.4228	0.2456	0.3969	0.0000	0.4752	3.4587	3.4745	0.8464
Grd192	0.9012	1.2567	3.5352	0.2389	0.4455	0.5873	0.4509	3.2039	3.6145	0.7619
Grd193	0.9178	2.0525	3.8618	0.2434	0.4400	0.6837	0.4522	3.3819	3.7472	0.6304
Grd194	1.3398	1.3492	3.3216	0.2248	0.4678	0.5603	0.4525	3.2793	3.9330	0.5657
Grd195	1.0896	2.0958	3.5304	0.2805	0.4757	0.9795	0.5987	3.3773	4.0857	0.7344
Grd196	0.9863	1.5879	3.3512	0.2520	0.4215	0.7689	0.4653	3.4097	3.6216	0.8007
Grd197	1.0981	2.3950	3.9630	0.2482	0.4124	0.7241	0.4525	3.3403	1.5578	0.7878
Grd198	0.8950	1.6392	3.7996	0.2528	0.4151	0.7761	0.4569	3.1585	3.5581	0.8002
Grd199	0.9541	1.7154	3.7664	0.2825	0.4717	0.7895	0.4825	3.6861	4.2609	0.5923
Grd200	0.9276	2.5915	3.7113	0.2510	0.4339	1.2713	0.4410	3.3191	3.7205	0.9803
Grd201	0.9529	1.1600	3.3406	0.2566	0.4288	0.6088	0.4339	3.3263	3.8152	0.5870
Grd202	1.2940	1.1883	3.3286	0.2413	0.4459	0.6870	0.4536	3.3114	3.9879	0.2799
Grd203	1.0100	2.2636	3.9279	0.2570	0.5154	1.0138	0.5190	3.3461	4.4104	0.7466
Grd204	0.8854	1.9522	3.8777	0.2655	0.5691	1.1363	0.5399	3.2287	4.4588	0.3101
Grd205	0.9319	1.6336	3.6327	0.2559	0.4226	0.7238	0.4819	3.0591	3.5373	0.0000
Grd206	0.9017	1.8755	3.8140	0.2434	0.3910	0.7499	0.4587	3.3260	3.7561	0.2978
Grd207	0.9449	2.2712	3.6584	0.2348	0.3722	0.7755	0.4687	3.2266	3.8316	0.7750
Grd208	1.4538	3.0206	4.0592	0.2839	0.4708	0.9874	0.5513	3.6427	4.5943	0.7304
Grd209	1.3086	1.4261	3.9163	0.2567	0.4080	0.7680	0.4582	3.0124	3.5412	0.7262
Grd210	0.9856	1.4301	3.5747	0.0000	0.4096	0.8763	0.4576	3.0634	0.0000	0.7494
Grd211	1.3168	2.3650	3.7048	0.2720	0.4373	0.6015	0.4748	3.2465	3.9079	0.6210
Grd212	0.8936	2.1543	3.1523	0.0000	0.4433	0.8375	0.4651	0.0000	3.8338	0.5957
Grd213	1.0305	1.5871	3.5399	0.2323	0.3813	0.7781	0.4485	2.9702	1.5786	0.7541
Grd214	1.0362	1.5687	4.3680	0.2691	0.5663	0.8506	0.6308	3.1526	5.1100	0.3235
Grd215	0.9109	1.2316	3.2378	0.2703	0.5633	0.9330	0.5048	4.0862	3.9735	1.0161
Grd216	1.0728	2.8146	3.4508	0.2505	0.3969	1.0570	0.4705	3.1753	3.6764	1.0998
Grd217	0.9873	1.2981	3.8374	0.2521	0.3977	0.7765	0.4561	3.2586	3.6111	1.8130
Grd218	1.0114	1.2722	3.4805	0.2209	0.3993	0.8101	0.4589	3.2856	3.6622	0.8782

Grd219	0.0000	1.2332	3.7432	0.2453	0.4352	0.4910	0.4655	3.1490	3.9674	0.6900
Grd220	0.8903	1.4684	3.9219	0.2314	0.4621	0.6241	0.4246	2.9054	3.5574	0.6687
Grd221	1.2550	1.3286	3.7127	0.2568	0.3888	0.6005	0.4125	3.1903	4.0665	0.6680
Grd222	0.9337	1.3069	3.6523	0.2349	0.4526	0.8829	0.4466	3.0296	4.0186	0.6549
Grd223	0.9873	3.6229	3.6585	0.2313	0.3799	0.7861	0.4561	3.2095	1.5121	0.6600
Grd224	0.9427	2.0733	3.5995	0.2444	0.0000	0.4696	0.4496	3.3046	3.8382	0.5805
Grd225	1.0267	2.2444	3.7619	0.2449	0.4090	0.6396	0.4608	3.3216	3.5935	0.7577
Grd226	0.8663	1.7848	3.8327	0.2482	0.3951	0.6835	0.0000	3.1411	3.6948	0.8821
Grd227	1.0095	1.6806	3.6593	0.2407	0.4069	1.1062	0.4663	3.2481	3.9715	1.5698
Grd228	0.9942	1.2708	3.5245	0.2598	0.4437	0.4664	0.4358	3.1557	3.7334	0.6650
Grd229	0.9876	1.9942	3.7339	0.2565	0.0000	0.8808	0.4666	3.3232	4.4235	0.5475
Grd230	1.0192	2.7734	4.1767	0.2707	0.5805	0.0000	0.4875	3.5260	4.4137	0.6433
Grd231	0.9594	1.5515	4.3249	0.2417	0.4945	0.7410	0.4729	3.1034	4.5225	0.6019
Grd232	0.9260	1.3906	3.7329	0.2476	0.3983	0.6090	0.4543	3.0733	3.8884	0.9483
Grd233	0.9269	1.1966	3.7557	0.2557	0.4091	0.6130	0.4771	3.2504	3.7263	1.1918
Grd234	0.9307	1.8035	3.7296	0.2511	0.4358	0.5618	0.4444	3.1920	3.8235	0.6941
Grd235	0.9159	1.5394	3.8077	0.2342	0.4713	0.5447	0.4415	3.2667	4.0743	0.6529
Grd236	0.9988	2.5288	3.5211	0.2324	0.4006	0.7078	0.4582	3.2194	3.5688	0.7910
Grd237	1.0475	1.5293	3.5066	0.0000	0.3920	0.8683	0.4536	3.0134	3.5674	0.2666
Grd238	0.9624	2.2677	3.7759	0.2462	0.4055	0.7196	0.4638	3.4865	4.0306	0.7545
Grd239	1.2551	1.2109	3.5599	0.2541	0.4312	0.6016	0.4442	3.0575	3.5530	0.6746
Grd240	1.3291	1.7226	3.8348	0.2697	0.4532	0.5980	0.4021	3.2376	3.7985	0.7132
Grd241	1.3831	1.3639	3.8664	0.2664	0.5320	0.0000	0.5237	3.1721	4.3526	0.7326
Grd242	1.0263	2.0717	3.4221	0.2571	0.3906	0.8850	0.4496	3.4369	3.4909	0.2725
Grd243	0.9258	1.8266	3.6216	0.2394	0.4040	0.7415	0.4474	0.0000	3.7385	1.6469
Grd244	0.9169	1.4003	3.8807	0.2489	0.4173	0.7564	0.4230	3.1186	3.7180	0.7972
Grd245	0.9365	1.1378	3.5986	0.2682	0.0000	1.2631	0.5179	3.2613	4.1006	0.7267
Grd246	0.9015	1.7019	3.8369	0.2562	0.4022	0.9223	0.4441	3.2799	3.5622	1.1178
Grd247	1.3654	1.9961	3.4206	0.2476	0.3977	0.8068	0.4766	3.1052	3.4051	0.7192
Grd248	1.1509	3.5823	3.4951	0.2517	0.4131	0.7686	0.4512	3.2555	3.6273	1.0773
Grd249	0.9247	2.4811	3.3639	0.2585	0.3877	0.8122	0.4476	3.0347	3.6856	0.7288
Grd250	1.3881	1.7870	3.6482	0.2529	0.5613	0.7915	0.5252	3.8259	0.0000	0.0000
Grd251	1.0162	1.5317	4.1748	0.2740	0.5567	0.0000	0.5241	3.3786	4.0856	0.7682
Grd252	0.9548	3.0605	4.2118	0.2705	0.5111	0.9997	0.4926	3.6109	4.6699	0.8329
Grd253	0.9241	1.6589	3.3866	0.2567	0.4054	0.6896	0.4718	3.1861	3.9892	0.8079
Grd254	0.9220	1.2297	3.5282	0.2465	0.4404	0.7417	0.3956	3.2799	0.0000	0.6497
Grd255	1.1738	1.3676	3.6335	0.2599	0.3871	0.7276	0.4320	3.3175	3.5876	0.7877
Grd256	1.0815	1.5521	3.6229	0.2449	0.3796	0.8352	0.4615	3.0589	3.7033	0.9581
Grd257	1.0203	1.4950	3.4629	0.2346	0.4230	0.8156	0.4454	3.3512	3.6125	0.8219
Grd258	0.9924	1.8617	3.6819	0.2673	0.4856	0.9388	0.6059	3.5462	4.1591	0.6599
Grd259	0.9115	3.9937	3.6154	0.2423	0.4177	0.9061	0.4504	0.0000	3.2042	1.6930
Grd260	1.2827	1.5969	3.6910	0.2505	0.4062	0.7167	0.4563	3.2005	3.5098	0.8032
Grd261	1.3011	1.4447	3.6055	0.2624	0.4251	0.7345	0.4233	3.3619	3.4549	0.2725
Grd262	0.9531	1.1692	3.6392	0.2602	0.4069	0.7975	0.4235	0.0000	1.5684	0.2903
Grd263	0.9719	3.0566	3.8350	0.2535	0.4114	0.7686	0.4375	3.0916	0.0000	0.8461

Grd264	0.8850	1.2311	4.6537	0.3033	0.4788	1.0327	0.6746	3.1939	4.3856	0.5758
Grd265	1.1017	1.4006	3.4437	0.2609	0.3971	0.7585	0.4823	3.2679	3.9902	0.2879
Grd266	1.3723	2.0982	3.7034	0.2444	0.4210	0.7576	0.4623	3.1778	3.8633	1.0323
Grd267	0.9005	1.3111	3.5302	0.2967	0.5129	0.9225	0.5511	3.1152	3.9936	0.8421
Grd268	0.9943	1.6786	3.5423	0.2414	0.4248	0.8878	0.4596	3.3401	3.8716	0.7123
Grd269	1.4105	1.3812	4.4631	0.2595	0.4888	1.0573	0.6237	3.6528	4.2717	0.7105
Grd270	0.9246	1.1848	3.4859	0.2518	0.3695	1.5245	0.4386	3.2602	3.9613	0.8191
Grd271	0.9562	2.0671	3.7790	0.2498	0.3993	0.7262	0.4330	3.1408	3.8540	0.7798
Grd272	1.3816	2.3325	3.5041	0.2481	0.3893	0.7373	0.4624	3.3289	3.5502	0.8290
Grd273	1.0738	1.3266	3.4330	0.2577	0.4029	0.7419	0.4581	3.4594	3.8040	0.8326
Grd274	1.5223	1.7042	4.3337	0.2677	0.5301	0.9575	0.5251	3.1629	4.5473	0.6690
Grd275	1.3273	2.6196	3.8578	0.2675	0.5133	0.9270	0.5550	3.3345	4.5010	0.7493
Grd276	0.9584	1.0669	4.0142	0.2591	0.4139	0.7719	0.4900	3.4429	4.1989	0.2998
Grd277	1.0575	2.0105	3.6598	0.2617	0.3896	1.5316	0.4476	3.4100	3.6509	0.8663
Grd278	0.9233	0.0000	3.2108	0.2436	0.5079	0.9847	0.4766	3.3441	3.6714	0.7610
Grd279	1.0049	1.3245	3.4371	0.0000	0.3895	0.8293	0.4623	3.1601	3.8164	0.8372
Grd280	1.0288	1.4920	3.6133	0.2494	0.4071	0.7929	0.4263	2.9451	3.9210	0.8284
Grd281	1.4614	2.6651	4.1524	0.2461	0.5242	0.9022	0.5955	3.6482	0.0000	0.7238
Grd282	0.9119	1.8995	3.9533	0.2646	0.4217	0.7436	0.4441	3.3117	3.7817	0.6958
Grd283	1.8988	2.1427	3.7097	0.2435	0.3891	1.8906	0.4477	3.3327	3.4205	0.7914
Grd284	0.8896	2.8861	3.8537	0.2452	0.5308	0.8674	0.4454	3.1562	4.7198	0.8072
Grd285	0.9604	1.7335	3.6383	0.2441	0.3910	1.1625	0.4634	3.2310	3.6586	0.8229
Grd286	0.8872	3.0381	3.7999	0.2596	0.4379	0.6035	0.4405	3.1734	3.8136	0.7183
Grd287	1.2915	2.1138	3.4829	0.2537	0.4621	0.6303	0.4452	3.3005	4.0055	0.7282
Grd288	0.9287	1.4893	3.6644	0.2550	0.4394	0.6661	0.4425	3.3044	3.9252	0.7458
Grd289	0.9138	1.4647	3.9657	0.2569	0.4270	0.6364	0.4405	0.0000	4.2543	0.6260
Grd290	0.9784	2.5169	3.6086	0.2521	0.4285	0.8671	0.4497	3.0834	3.8967	0.6525
Grd291	1.0391	2.5302	4.4298	0.2508	0.5699	1.0082	0.4976	3.4648	4.0656	0.5866
Grd292	0.9113	2.4117	3.6850	0.2764	0.5166	0.9435	0.5173	3.3503	3.9781	0.8731
Grd293	0.9969	1.3725	3.4578	0.2269	0.4348	0.7128	0.4340	3.2910	4.0734	0.7161
Grd294	1.2158	2.2635	3.6015	0.2459	0.3922	0.7798	0.4355	3.4980	3.4263	0.7704
Grd295	1.0853	1.8665	3.6433	0.2629	0.3803	0.7312	0.4742	3.3130	0.0000	0.8488
Grd296	0.9150	2.3572	4.2421	0.2493	0.4285	0.7200	0.4391	3.2843	3.6281	1.0242
Grd297	1.3563	1.2692	3.3694	0.2589	0.3976	1.3665	0.4661	3.3743	3.5733	0.8260
Grd298	1.3478	1.8375	3.3871	0.2639	0.4339	0.8687	0.4559	3.2969	3.8000	0.2789
Grd299	0.9969	1.5832	4.9118	0.2411	0.5298	0.9203	0.5379	3.3601	4.7663	0.7726
Grd300	0.8722	2.8065	3.6641	0.2342	0.4673	0.9324	0.5117	3.5533	3.8924	0.7633
Grd301	0.0000	1.8215	3.5969	0.2381	0.3850	0.7652	0.4336	3.3376	3.4900	1.6742
Grd302	1.0065	1.4016	3.6099	0.2522	0.0000	1.1595	0.4528	3.5814	3.7855	0.9266
Grd303	1.0293	2.0701	4.4658	0.2545	0.5279	1.2069	0.6490	3.7617	4.6356	0.7169
Grd304	0.9809	1.2462	3.5817	0.2425	0.4443	0.6340	0.3818	3.2691	3.7493	0.7282
Grd305	1.1045	1.4106	3.6995	0.2618	0.0000	0.6919	0.4613	3.1942	3.9739	0.7514
Grd306	1.0140	1.6669	3.4635	0.2648	0.4018	1.8332	0.4903	3.1471	3.8572	0.8495
Grd307	0.9432	2.0345	3.5785	0.2335	0.4052	0.7064	0.4285	3.4113	3.9906	0.2962
Grd308	1.1681	1.3857	3.6397	0.2290	0.4129	0.0000	0.4578	3.1259	3.7092	0.7161

HB1	0.9138	4.4024	4.3968	0.2521	0.4344	0.8911	0.4491	3.2972	2.9117	0.9280
HB2	1.1320	3.5612	4.4811	0.2432	0.3580	0.8450	0.4506	3.0419	3.3527	0.9555
HB3	0.9269	7.4568	4.3252	0.2659	0.3820	0.7071	0.4799	3.0799	1.5691	1.2550
HB4	0.9877	3.3876	3.7512	0.2536	0.3964	0.6685	0.4406	3.1936	3.7442	0.7699
HB5	0.8962	4.7641	4.1804	0.2342	0.3925	0.9697	0.4195	2.8018	1.5313	0.9658
HB6	0.9267	4.8024	3.9612	0.2482	0.3889	0.6206	0.5134	3.3241	1.5329	1.1554
HB7	0.9553	5.4053	3.8898	0.2487	0.3913	0.5693	0.4670	3.0825	1.4739	1.1050
HB8	0.9659	2.5799	4.2665	0.2489	0.4004	0.6212	0.4671	3.1099	3.0722	0.7928
HB9	1.0791	3.2337	4.7668	0.2912	0.4563	0.5400	0.4645	4.5554	3.3096	1.1491
HB10	0.9238	2.9086	4.4039	0.2552	0.4665	0.4883	0.4659	3.1362	3.2510	0.6673
HB11	0.9797	4.2323	4.3227	0.2705	0.3700	0.7300	0.4841	3.1629	2.9863	1.2692
HB12	1.2509	6.9374	4.1049	0.3295	0.4695	0.6656	0.4028	3.2743	3.7238	1.0926
HB13	0.9812	9.4351	4.8708	0.2524	0.3899	0.6933	0.4701	3.1465	1.4287	1.0723
Rdm1	0.9725	4.4966	5.5431	0.2511	0.5273	0.5056	0.4873	3.8525	3.5568	0.9342
Rdm2	0.9701	4.5530	4.2698	0.2401	0.3547	0.7003	0.0000	3.0369	1.5721	1.0173
Rdm3	1.1570	2.6678	3.9518	0.2323	0.4150	0.6927	0.4693	3.1344	3.4488	0.6482
Rdm4	0.9585	5.7842	3.9735	0.2435	0.5241	0.8948	0.5110	3.4552	4.1994	1.0062
Rdm5	0.9311	2.2299	4.1145	0.2430	0.3983	0.6585	0.4476	3.2836	3.8519	0.9444
Rdm6	0.9582	5.5148	5.0571	0.2303	0.5562	0.7381	0.4627	3.6327	4.1251	1.1523
Rdm7	1.0870	3.1483	3.9239	0.2493	0.4038	0.6926	0.4591	3.1880	3.2150	1.0158
Rdm8	0.8800	5.4025	4.4105	0.2205	0.4068	0.4014	0.5119	3.5652	3.3674	0.9953
Rdm9	0.9240	5.8709	3.7288	0.2362	0.4340	0.5822	0.4554	3.2185	3.9133	0.9320
Rdm10	1.0146	4.8588	3.6053	0.2761	0.3635	0.8039	0.0000	2.7166	1.4745	1.0444
Rdm11	1.1585	5.8920	4.1040	0.2318	0.3751	0.7231	0.4775	2.8996	1.5532	1.0229
Rdm12	0.8538	4.1575	3.7503	0.2370	0.4664	0.6035	0.3870	3.4936	3.6947	0.6889
Rdm13	0.9555	2.8956	4.0682	0.2465	0.4198	0.6850	0.4695	3.9547	3.1886	1.0074
Rdm14	1.0500	4.7672	3.9847	0.2363	0.3772	0.6923	0.4512	3.2646	1.5835	1.0063
Rdm15	0.9545	4.0243	4.4469	0.2454	0.3649	0.6069	0.5003	3.5582	3.1538	0.8600
Rdm16	0.9521	5.8499	4.6452	0.2346	0.4693	0.5187	0.4777	3.5779	3.1333	0.8835
Rdm17	0.9029	3.7426	5.6012	0.2447	0.5504	0.8722	0.0000	3.4209	4.6992	0.3305
Rdm18	0.9650	3.6309	3.3873	0.2721	0.4085	0.7009	0.4689	3.1634	1.4632	1.0878
Rdm19	1.0796	2.4110	4.3663	0.2850	0.4439	0.6849	0.4821	4.0998	3.5179	1.1074
Rdm20	0.9940	3.3759	4.8715	0.2620	0.4329	0.6688	0.4755	3.3817	2.8790	1.0061
Rdm21	0.9547	3.1324	5.0580	0.2451	0.4354	0.6482	0.0000	2.7107	3.5358	1.0484
Rdm22	0.8964	2.0151	4.5181	0.2567	0.5550	0.5865	0.6745	3.5707	4.0700	1.1622
Rdm23	1.0923	3.2968	5.3857	0.2649	0.4374	0.6869	0.4177	3.5464	3.3779	1.4526
Rdm24	1.0303	5.2761	3.4944	0.3234	0.4665	0.6866	0.5604	3.6836	3.7740	1.2108
Rdm25	1.0288	4.8328	5.3079	0.2835	0.4930	0.6245	0.5824	3.5518	4.2185	1.2377
Rdm26	0.9427	3.5963	4.1387	0.2415	0.4419	0.6195	0.4350	3.6188	3.2960	1.0031
Rdm27	0.0000	3.4623	4.4650	0.0000	0.4509	0.5182	0.3980	4.0016	3.9599	1.0729

Appendix F: Calculated rates of Optimality, Proximity and Under-performance for each configuration candidate tested on shortened sequences

Configuration Name	Δo (%)	Δp (%)	Δu (%)
Gen1	50	20	30
Gen2	50	0	50
Gen3	40	0	40
Gen4	60	20	20
Gen5	70	20	10
Gen6	50	20	20
Gen7	50	20	30
Gen8	60	20	20
Gen9	70	0	30
Gen10	40	20	40
Gen11	50	30	20
Gen12	70	20	10
Gen13	20	10	60
Gen14	50	30	20
Gen15	60	10	20
Gen16	60	20	20
Gen17	50	10	40
Gen18	70	10	20
Gen19	30	50	20
Gen20	40	10	40
Gen21	90	0	10
Gen22	70	0	30
Gen23	60	20	20
Gen24	70	0	30
Gen25	70	10	20
Gen26	50	20	30
Gen27	70	10	20
Gen28	60	30	10
Gen29	50	20	30
Gen30	70	10	20
Gen31	40	0	60
Gen32	70	0	30
Gen33	60	0	30
Gen34	50	10	30
Gen35	60	10	30
Gen36	60	10	30
Gen37	50	40	10
Gen38	40	40	20
Gen39	60	0	40
Gen40	20	30	50
Gen41	60	10	20
Gen42	30	20	40
Gen43	40	20	30

Gen44	40	20	20
Gen45	50	10	30
Gen46	50	20	20
Gen47	50	20	30
Gen48	30	10	50
Gen49	60	20	10
Gen50	50	20	20
Gen51	80	10	10
Gen52	50	0	50
Gen53	50	30	20
Gen54	50	20	30
Gen55	40	0	50
Gen56	60	10	30
Gen57	50	10	40
Gen58	40	10	50
Gen59	50	0	50
Gen60	30	20	50
Gen61	40	0	60
Gen62	60	0	40
Gen63	50	0	50
Gen64	60	30	10
Gen65	40	20	40
Gen66	60	0	40
Gen67	50	0	40
Gen68	80	10	10
Gen69	50	0	40
Gen70	20	40	30
Gen71	70	10	10
Gen72	50	10	30
Gen73	60	10	30
Gen74	30	10	50
Gen75	50	20	30
Gen76	70	20	10
Gen77	40	30	30
Gen78	60	0	40
Gen79	60	0	30
Gen80	60	0	40
Gen81	60	10	30
Gen82	40	10	30
Gen83	40	0	50
Gen84	50	20	30
Gen85	50	10	40
Gen86	40	0	60
Gen87	40	0	40

Gen88	60	10	30
Gen89	60	10	30
Gen90	40	40	20
Gen91	60	10	30
Gen92	50	0	50
Gen93	50	20	30
Gen94	40	0	50
Gen95	60	20	20
Gen96	50	20	20
Gen97	40	20	30
Gen98	70	0	20
Gen99	60	10	30
Gen100	70	20	10
Gen101	50	20	30
Gen102	60	0	30
Gen103	50	10	40
Gen104	60	0	40
Gen105	60	20	20
Gen106	70	0	20
Gen107	40	30	30
Gen108	50	0	50
Gen109	30	10	50
Gen110	50	0	50
Gen111	50	30	20
Gen112	60	20	20
Gen113	40	10	50
Gen114	50	10	30
Gen115	30	30	30
Gen116	60	0	30
Gen117	50	0	50
Gen118	30	10	50
Gen119	40	0	40
Gen120	70	0	30
Gen121	60	10	20
Gen122	70	0	30
Gen123	80	0	20
Gen124	40	20	40
Gen125	50	10	40
Gen126	60	0	40
Gen127	40	10	40
Gen128	60	10	30
Gen129	40	0	50
Gen130	50	0	40
Gen131	60	20	20

Gen132	60	0	40
Gen133	50	40	10
Gen134	70	0	30
Gen135	50	10	40
Gen136	70	0	30
Gen137	50	0	30
Gen138	40	10	40
Gen139	50	0	50
Gen140	50	20	20
Gen141	70	10	20
Gen142	50	10	40
Gen143	60	20	20
Gen144	60	20	20
Gen145	70	20	10
Gen146	60	20	20
Gen147	70	0	30
Gen148	60	0	30
Gen149	60	10	30
Gen150	60	20	20
Gen151	60	0	30
Gen152	70	10	20
Gen153	40	40	20
Gen154	70	0	30
Gen155	70	0	30
Gen156	50	10	20
Gen157	30	10	50
Gen158	50	20	20
Gen159	40	0	50
Gen160	80	0	20
Gen161	60	10	10
Gen162	50	40	10
Gen163	40	10	40
Gen164	50	10	40
Gen165	50	30	20
Gen166	70	10	20
Gen167	40	0	60
Gen168	30	10	60
Gen169	60	20	20
Gen170	70	0	20
Gen171	40	30	20
Gen172	50	10	40
Gen173	40	30	30
Gen174	70	10	20
Gen175	50	10	40

Gen176	40	0	50
Gen177	50	10	30
Gen178	60	10	30
Gen179	40	20	40
Gen180	60	10	30
Gen181	60	0	40
Gen182	60	20	20
Gen183	70	10	20
Gen184	20	10	60
Gen185	60	10	30
Grd1	40	10	50
Grd2	30	20	50
Grd3	60	10	30
Grd4	60	0	20
Grd5	30	0	50
Grd6	30	10	60
Grd7	50	10	40
Grd8	60	0	40
Grd9	40	10	50
Grd10	30	0	70
Grd11	40	0	60
Grd12	70	10	20
Grd13	30	0	70
Grd14	40	10	50
Grd15	30	10	60
Grd16	70	0	30
Grd17	40	20	40
Grd18	40	20	30
Grd19	40	0	60
Grd20	30	20	50
Grd21	40	0	60
Grd22	30	10	60
Grd23	60	10	30
Grd24	50	0	50
Grd25	60	20	20
Grd26	40	10	50
Grd27	50	10	40
Grd28	30	30	40
Grd29	60	0	30
Grd30	30	10	60
Grd31	40	10	40
Grd32	60	20	20
Grd33	30	10	50
Grd34	40	10	50

Grd35	60	20	20
Grd36	50	20	30
Grd37	40	0	60
Grd38	50	20	30
Grd39	40	10	50
Grd40	70	10	20
Grd41	40	20	40
Grd42	30	10	60
Grd43	50	10	40
Grd44	40	0	60
Grd45	60	10	30
Grd46	40	30	30
Grd47	70	0	30
Grd48	70	10	20
Grd49	40	10	40
Grd50	50	10	30
Grd51	60	10	20
Grd52	50	20	20
Grd53	30	0	70
Grd54	30	0	60
Grd55	40	20	40
Grd56	40	10	50
Grd57	40	10	50
Grd58	40	10	50
Grd59	50	10	40
Grd60	40	10	50
Grd61	50	20	30
Grd62	40	0	60
Grd63	40	0	60
Grd64	60	10	30
Grd65	50	20	30
Grd66	40	0	60
Grd67	40	0	50
Grd68	40	20	40
Grd69	60	10	30
Grd70	50	10	40
Grd71	60	20	20
Grd72	30	0	70
Grd73	40	10	40
Grd74	50	10	40
Grd75	60	20	20
Grd76	40	0	50
Grd77	50	30	20
Grd78	30	0	70

Grd79	50	20	30
Grd80	90	0	10
Grd81	40	30	30
Grd82	40	10	40
Grd83	50	20	30
Grd84	40	10	50
Grd85	60	10	20
Grd86	60	0	40
Grd87	70	0	30
Grd88	40	10	50
Grd89	50	10	40
Grd90	40	10	50
Grd91	30	0	70
Grd92	60	20	20
Grd93	60	0	40
Grd94	60	30	10
Grd95	70	10	20
Grd96	40	20	40
Grd97	40	0	60
Grd98	50	0	50
Grd99	50	0	50
Grd100	40	10	40
Grd101	70	0	30
Grd102	30	0	70
Grd103	40	0	60
Grd104	70	0	30
Grd105	50	10	20
Grd106	50	10	40
Grd107	40	0	40
Grd108	50	0	40
Grd109	30	10	60
Grd110	50	20	20
Grd111	50	20	20
Grd112	30	10	60
Grd113	50	20	30
Grd114	50	10	40
Grd115	40	0	60
Grd116	60	20	20
Grd117	50	30	20
Grd118	60	0	40
Grd119	50	0	50
Grd120	60	10	30
Grd121	50	20	20
Grd122	50	10	40

Grd123	50	20	30
Grd124	50	40	10
Grd125	50	20	30
Grd126	70	0	30
Grd127	70	10	10
Grd128	60	10	20
Grd129	40	20	20
Grd130	80	0	20
Grd131	60	20	20
Grd132	60	20	20
Grd133	50	20	30
Grd134	50	20	30
Grd135	50	0	50
Grd136	30	10	50
Grd137	50	10	40
Grd138	60	10	30
Grd139	40	20	30
Grd140	60	10	30
Grd141	50	20	30
Grd142	50	20	30
Grd143	70	10	20
Grd144	40	10	50
Grd145	40	10	30
Grd146	70	10	20
Grd147	60	10	30
Grd148	20	20	30
Grd149	60	20	20
Grd150	70	10	20
Grd151	70	0	30
Grd152	40	30	30
Grd153	50	0	50
Grd154	40	10	40
Grd155	50	10	30
Grd156	30	0	70
Grd157	60	0	40
Grd158	60	10	30
Grd159	60	10	30
Grd160	60	0	40
Grd161	30	0	70
Grd162	40	10	50
Grd163	50	10	30
Grd164	40	20	40
Grd165	60	20	20
Grd166	60	10	20

Grd167	50	20	20
Grd168	50	20	30
Grd169	70	0	30
Grd170	40	0	60
Grd171	70	0	20
Grd172	60	10	30
Grd173	40	20	30
Grd174	30	20	40
Grd175	60	10	30
Grd176	50	20	30
Grd177	50	30	20
Grd178	70	0	30
Grd179	50	10	30
Grd180	70	10	20
Grd181	60	10	30
Grd182	40	10	30
Grd183	60	0	40
Grd184	70	0	30
Grd185	30	30	30
Grd186	60	20	20
Grd187	60	10	30
Grd188	50	10	40
Grd189	30	20	50
Grd190	40	30	30
Grd191	60	20	10
Grd192	80	0	20
Grd193	60	10	30
Grd194	70	0	30
Grd195	30	10	60
Grd196	40	40	20
Grd197	60	20	20
Grd198	70	10	20
Grd199	30	10	60
Grd200	50	20	30
Grd201	40	40	20
Grd202	60	0	40
Grd203	30	20	50
Grd204	50	0	50
Grd205	40	20	30
Grd206	70	10	20
Grd207	70	10	20
Grd208	30	0	70
Grd209	60	10	30
Grd210	60	10	10

Grd211	40	20	40
Grd212	40	10	30
Grd213	80	0	20
Grd214	40	0	60
Grd215	30	0	70
Grd216	50	10	40
Grd217	60	10	30
Grd218	70	0	30
Grd219	60	10	20
Grd220	70	10	20
Grd221	60	20	20
Grd222	70	0	30
Grd223	80	10	10
Grd224	80	0	10
Grd225	50	30	20
Grd226	70	0	20
Grd227	50	10	40
Grd228	60	10	30
Grd229	30	30	30
Grd230	30	0	60
Grd231	50	20	30
Grd232	70	20	10
Grd233	50	30	20
Grd234	80	0	20
Grd235	80	0	20
Grd236	70	0	30
Grd237	60	0	30
Grd238	50	30	20
Grd239	50	20	30
Grd240	60	0	40
Grd241	40	0	50
Grd242	50	20	30
Grd243	60	0	30
Grd244	70	10	20
Grd245	50	0	40
Grd246	60	10	30
Grd247	60	10	30
Grd248	50	10	40
Grd249	70	10	20
Grd250	30	0	50
Grd251	30	10	50
Grd252	30	10	60
Grd253	60	20	20
Grd254	70	0	20

Grd255	50	20	30
Grd256	50	20	30
Grd257	50	20	30
Grd258	30	0	70
Grd259	50	10	30
Grd260	70	0	30
Grd261	40	20	40
Grd262	60	20	10
Grd263	60	20	10
Grd264	50	0	50
Grd265	50	10	40
Grd266	40	20	40
Grd267	50	0	50
Grd268	40	30	30
Grd269	30	10	60
Grd270	80	0	20
Grd271	70	10	20
Grd272	50	20	30
Grd273	50	20	30
Grd274	40	0	60
Grd275	30	10	60
Grd276	30	40	30
Grd277	50	20	30
Grd278	40	20	30
Grd279	50	10	30
Grd280	70	0	30
Grd281	40	0	50
Grd282	60	10	30
Grd283	60	10	30
Grd284	70	0	30
Grd285	60	20	20
Grd286	60	20	20
Grd287	60	10	30
Grd288	60	20	20
Grd289	50	30	10
Grd290	60	10	30
Grd291	40	10	50
Grd292	40	10	50
Grd293	60	0	40
Grd294	60	0	40
Grd295	50	10	30
Grd296	60	0	40
Grd297	40	30	30
Grd298	50	0	50

Grd299	40	10	50
Grd300	50	0	50
Grd301	50	10	30
Grd302	40	10	40
Grd303	30	10	60
Grd304	60	20	20
Grd305	40	20	30
Grd306	50	0	50
Grd307	70	10	20
Grd308	60	10	20
HB1	60	10	30
HB2	60	10	30
HB3	50	10	40
HB4	70	10	20
HB5	80	10	10
HB6	60	20	20
HB7	60	30	10
HB8	60	30	10
HB9	30	10	60
HB10	60	20	20
HB11	40	10	50
HB12	30	10	60
HB13	50	20	30
Rdm1	40	20	40
Rdm2	60	10	20
Rdm3	50	20	30
Rdm4	20	40	40
Rdm5	70	20	10
Rdm6	20	30	50
Rdm7	60	0	40
Rdm8	50	20	30
Rdm9	60	20	20
Rdm10	50	0	40
Rdm11	50	20	30
Rdm12	60	10	30
Rdm13	30	40	30
Rdm14	70	10	20
Rdm15	50	20	30
Rdm16	40	30	30
Rdm17	50	10	30
Rdm18	50	20	30
Rdm19	20	0	80
Rdm20	20	50	30
Rdm21	40	20	30

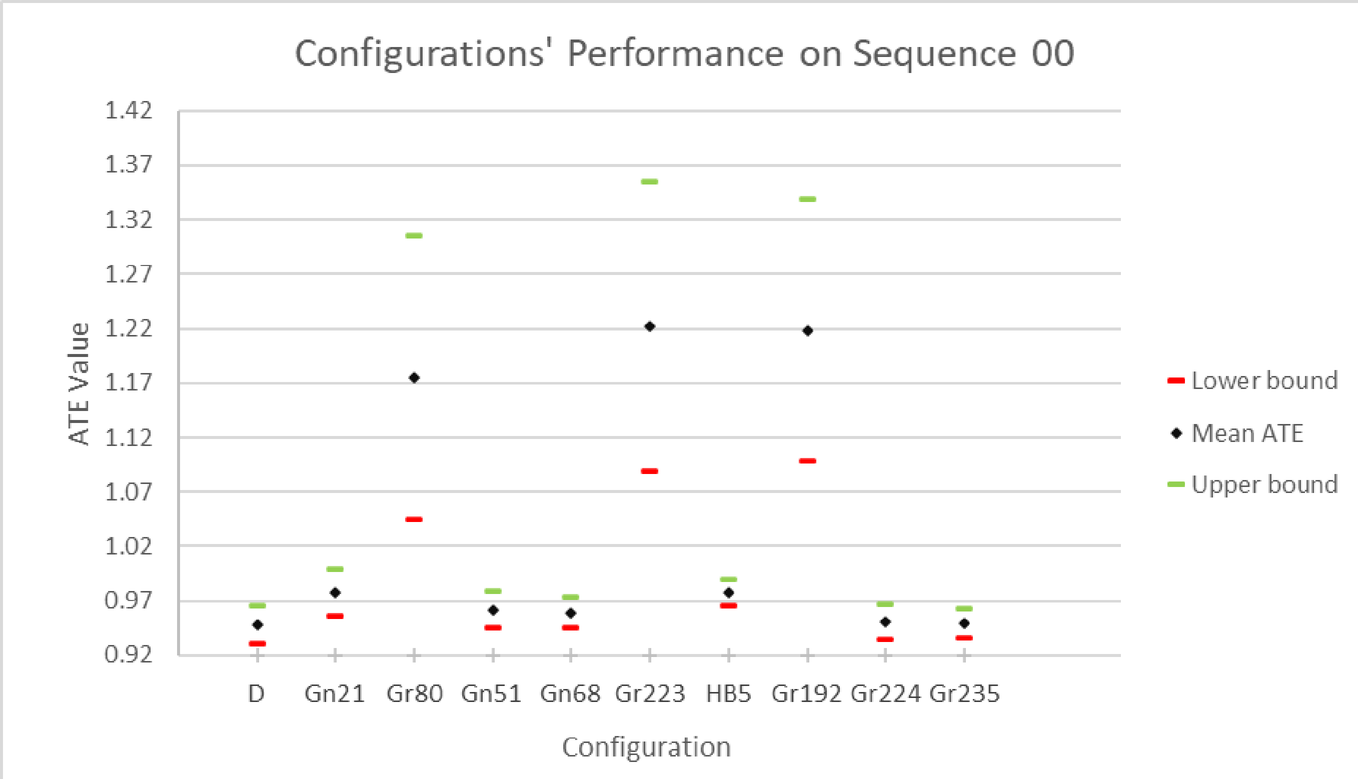
Rdm22	40	10	50
Rdm23	30	0	70
Rdm24	10	10	80
Rdm25	20	10	70
Rdm26	50	20	30
Rdm27	40	0	40

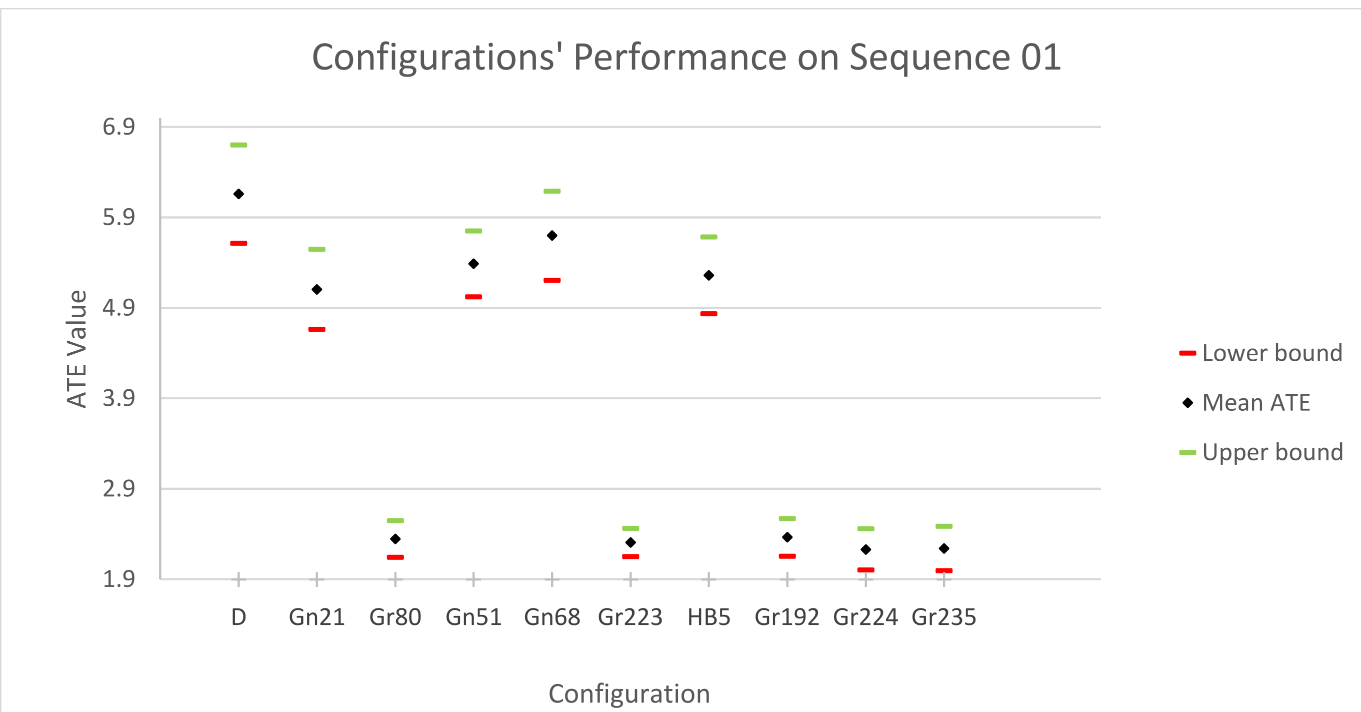
Appendix G: Full Sequence Test Results: ATE evaluation per testing sequence

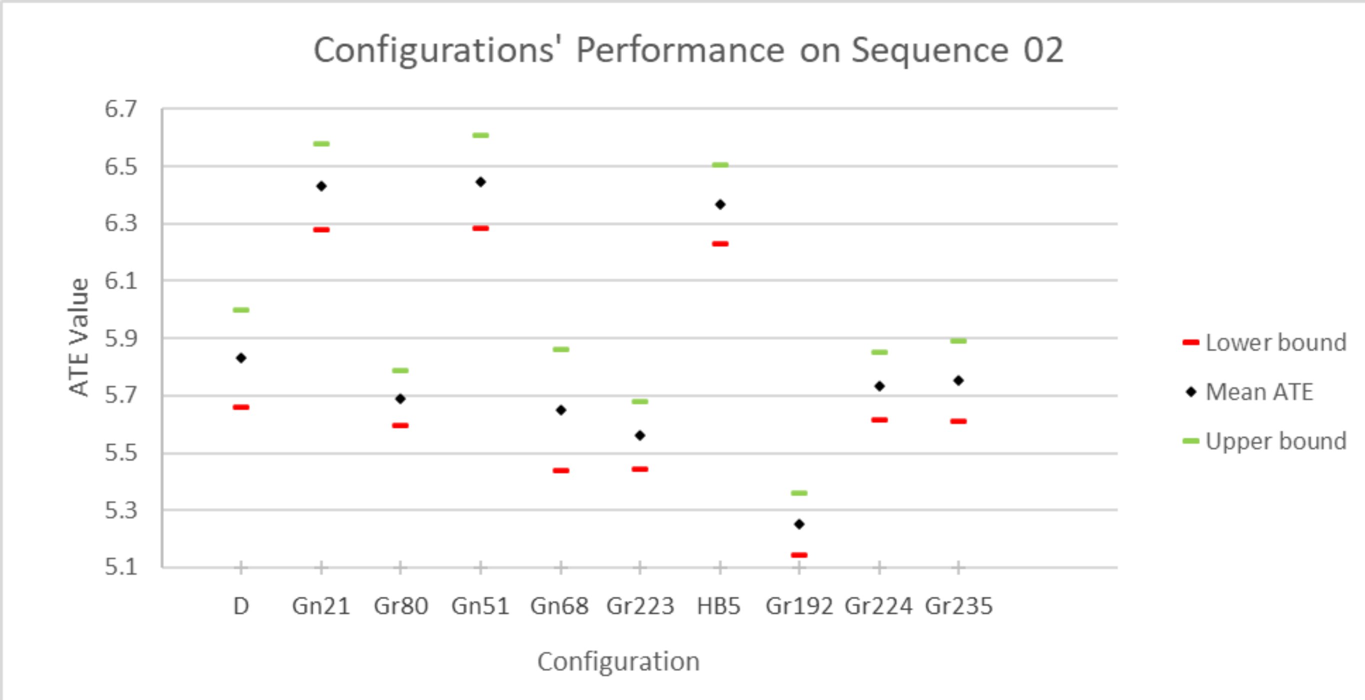
Table G.1: ATE calculated for each configuration candidate on each of the testing sequences.

Configuration	**ATE calculated per sequence**									
Name	S00	S01	S02	S03	S05	S06	S07	S08	S09	S10
Gen21	0.9776	5.1061	6.4299	0.3432	0.3898	0.7789	0.4447	3.1264	1.6922	1.0569
Grd80	1.1751	2.3464	5.6898	0.3036	0.3688	0.7861	0.4337	3.1776	3.5077	0.9753
Gen51	0.9614	5.3881	6.4451	0.3226	0.4053	0.7610	0.4445	3.1387	1.6831	1.0667
Gen68	0.9592	5.6997	5.6489	0.3392	0.4319	0.7827	0.4781	3.1082	1.5248	1.1006
Grd223	1.2220	2.3093	5.5624	0.3034	0.3715	0.8651	0.4404	3.2048	3.4389	0.8095
HB5	0.9770	5.2590	6.3669	0.3555	0.4139	0.7504	0.4555	3.0857	1.5484	1.1051
Grd192	1.2185	2.3648	5.2516	0.2940	0.3852	0.8476	0.4243	3.2351	3.5685	0.8166
Grd224	0.9506	2.2315	5.7335	0.2908	0.3897	0.7342	0.4180	3.2627	3.7906	0.7200
Grd235	0.9489	2.2405	5.7514	0.2895	0.3870	0.8235	0.4285	3.2337	3.6083	0.7857

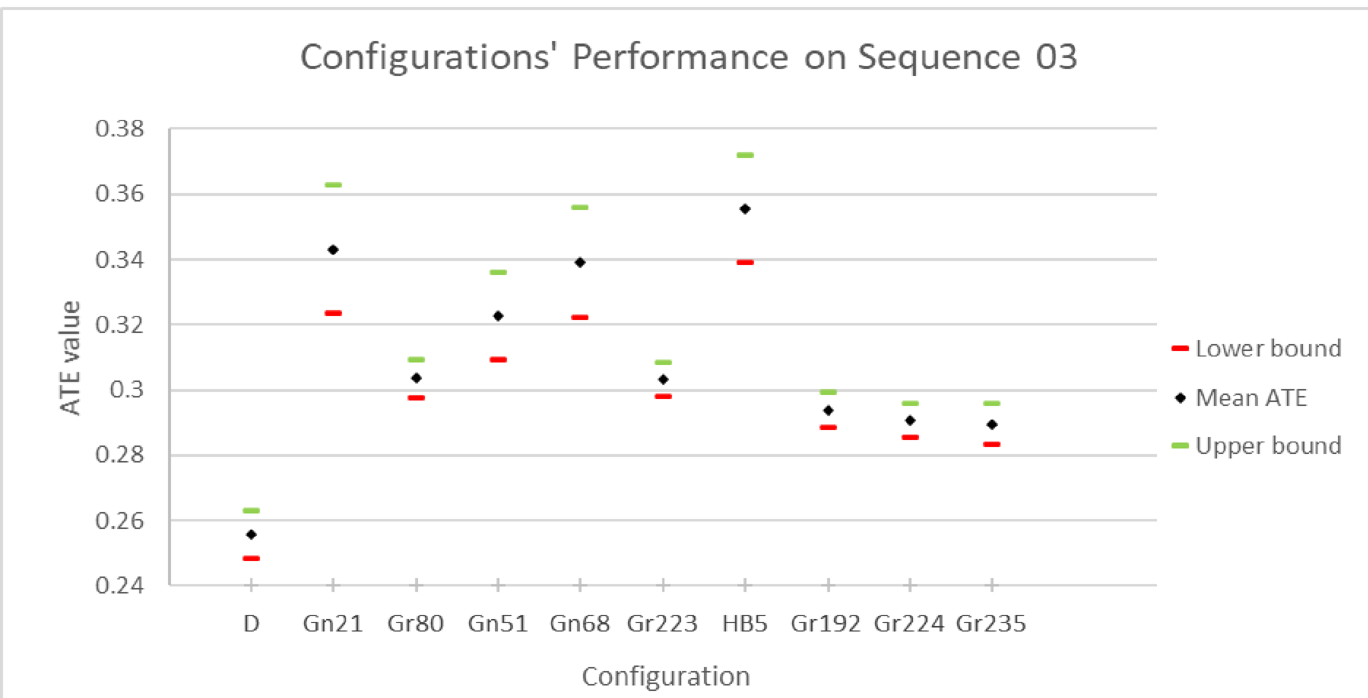
Appendix H: Configurations' Performance on each of the KITTI Sequences


Configurations' Performance on Sequence 00
ATE Value
1.42
1.37
1.32
1.27
1.22
1.17
1.12
1.07
1.02
0.97
0.92
D
Gn21
Gr80
Gn51
Gn68
Gr223
HB5
Gr192
Gr224
Gr235
Configuration
Lower bound
Mean ATE
Upper bound

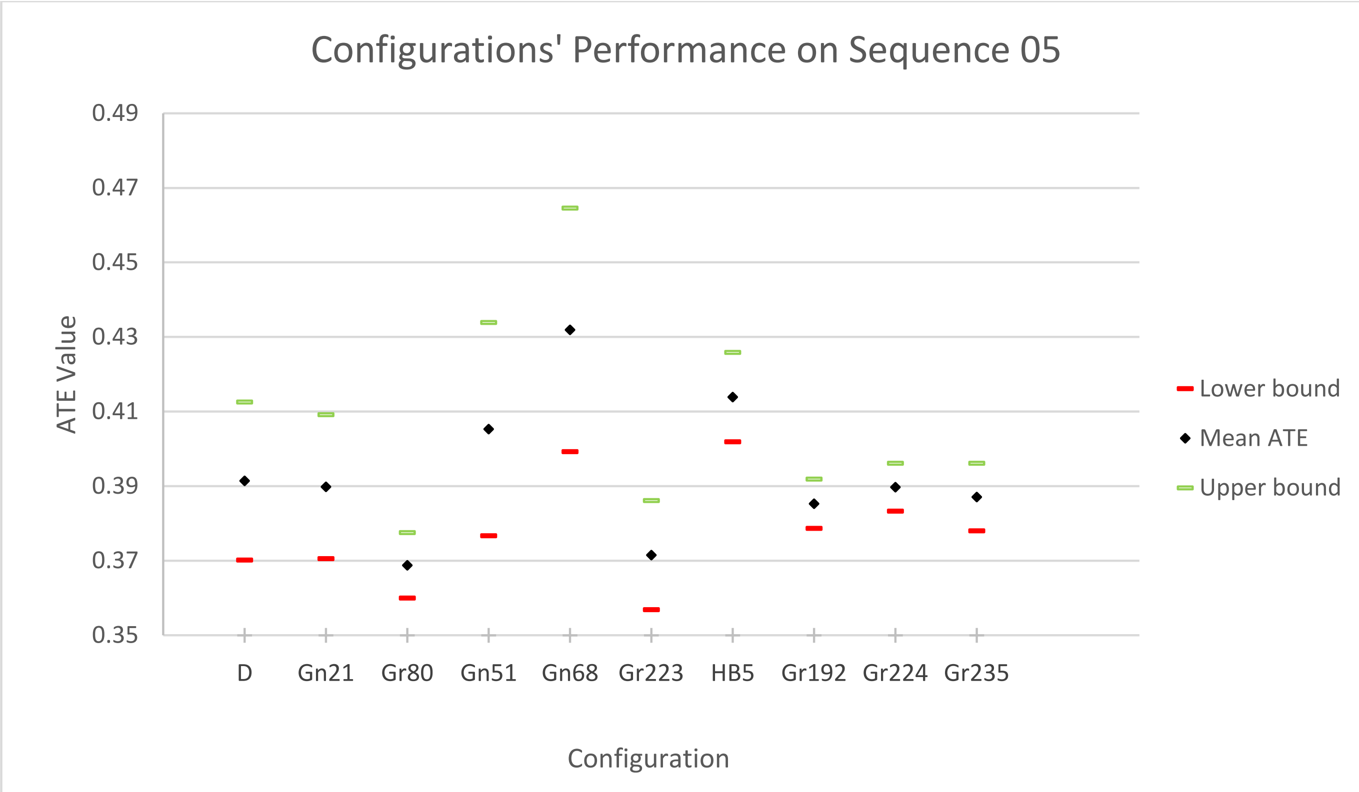

Configurations' Performance on Sequence 01
ATE Value
6.9
5.9
4.9
3.9
2.9
1.9
D
Gn21
Gr80
Gn51
Gn68
Gr223
HB5
Gr192
Gr224
Gr235
Configuration
Lower bound
Mean ATE
Upper bound

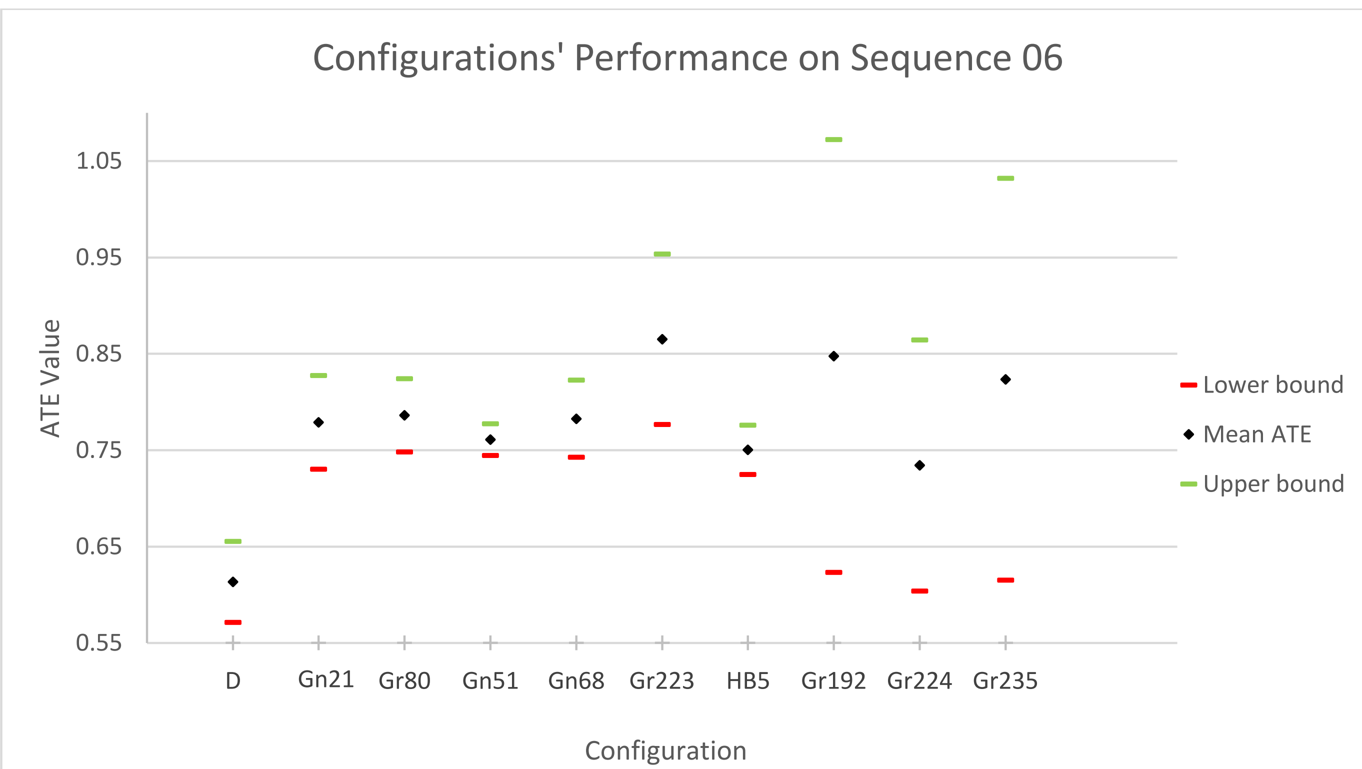


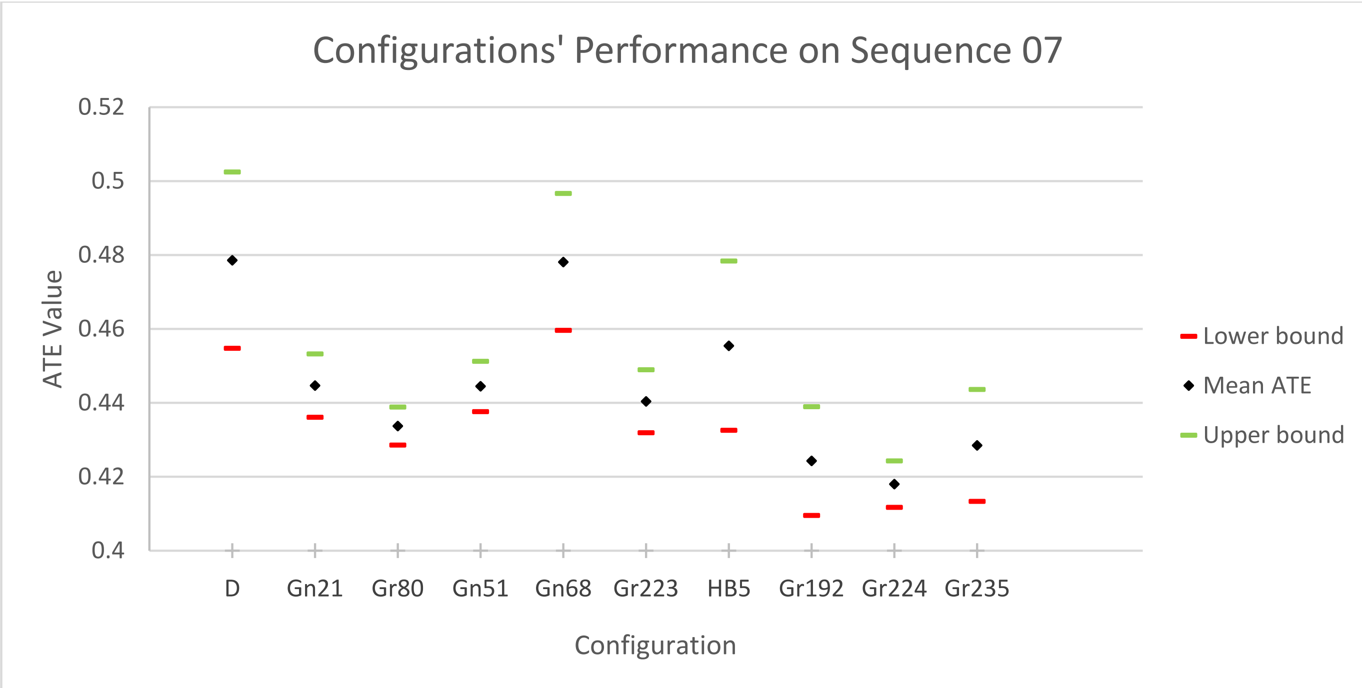
Configurations' Performance on Sequence 02
ATE Value
6.7
6.5
6.3
6.1
5.9
5.7
5.5
5.3
5.1
D
Gn21
Gr80
Gn51
Gn68
Gr223
HB5
Gr192
Gr224
Gr235
Configuration
Lower bound
Mean ATE
Upper bound

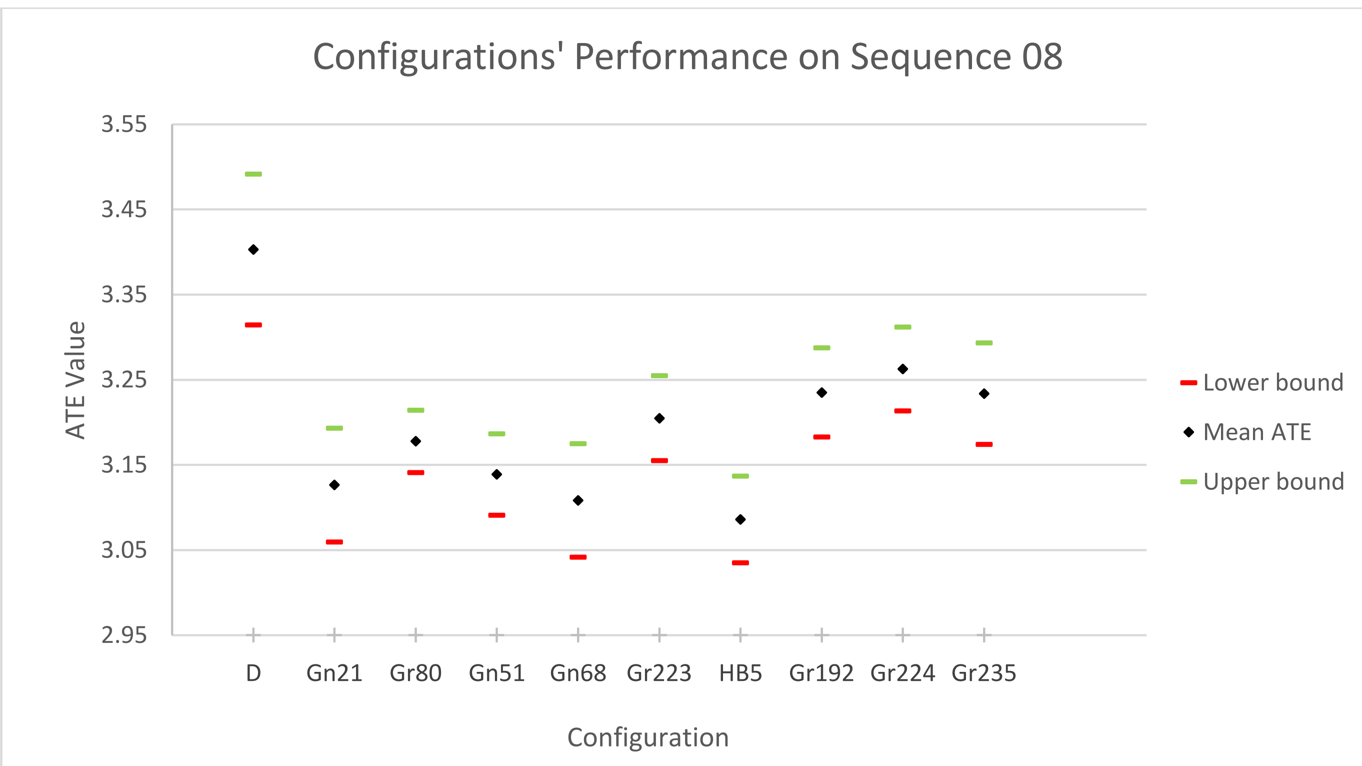


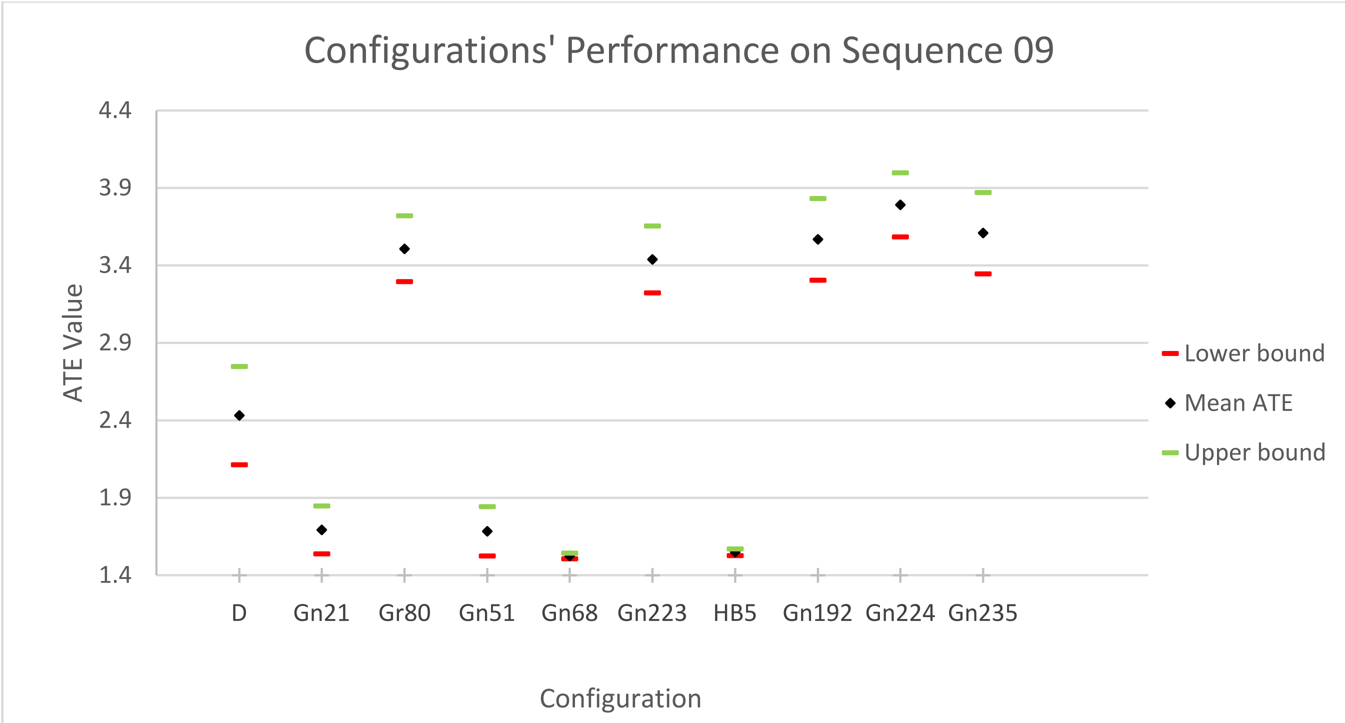
Configurations' Performance on Sequence 03
ATE value
0.38
0.36
0.34
0.32
0.3
0.28
0.26
0.24
D
Gn21
Gr80
Gn51
Gn68
Gr223
HB5
Gr192
Gr224
Gr235
Configuration
Lower bound
Mean ATE
Upper bound

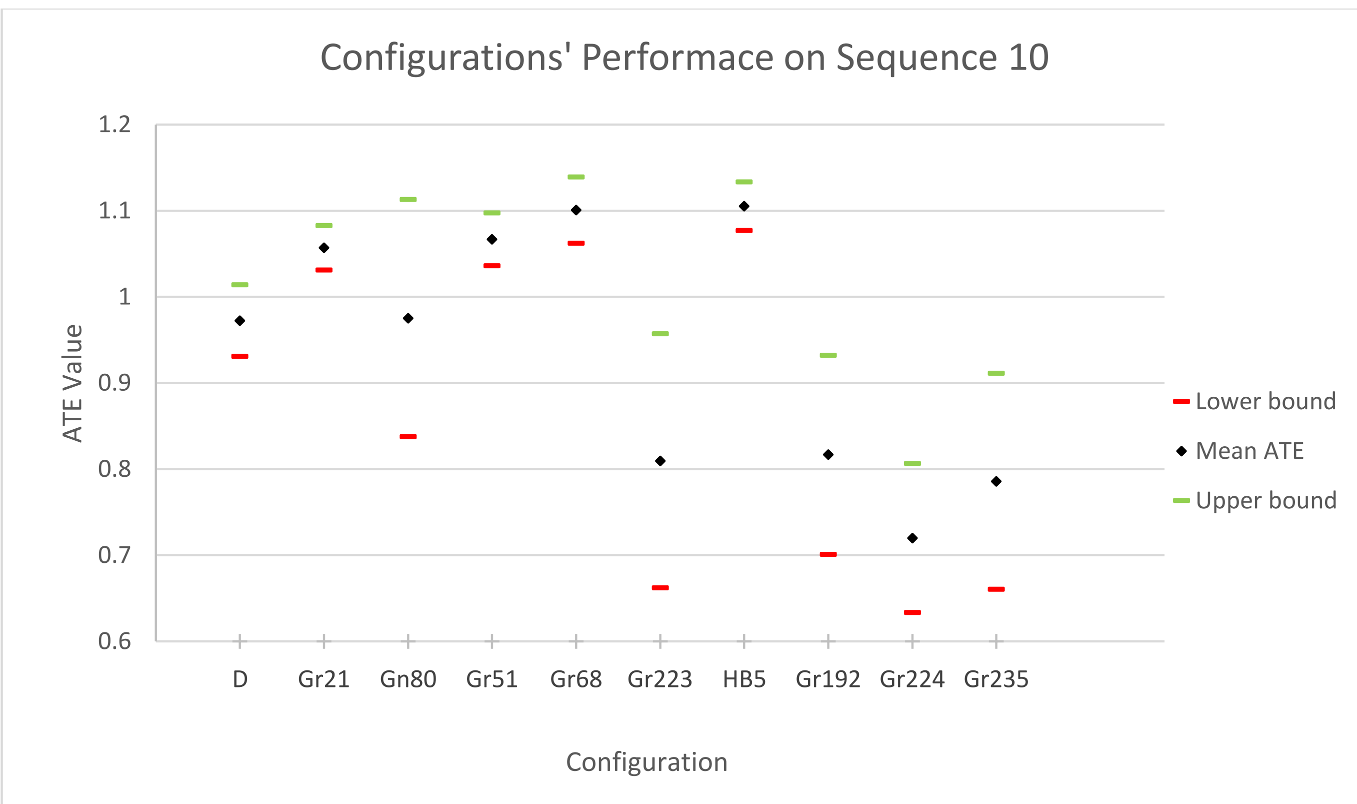

Configurations' Performance on Sequence 05
ATE Value
0.49
0.47
0.45
0.43
0.41
0.39
0.37
0.35
D
Gn21
Gr80
Gn51
Gn68
Gr223
HB5
Gr192
Gr224
Gr235
Configuration
Lower bound
Mean ATE
Upper bound


Configurations' Performance on Sequence 06
ATE Value
1.05
0.95
0.85
0.75
0.65
0.55
D
Gn21
Gr80
Gn51
Gn68
Gr223
HB5
Gr192
Gr224
Gr235
Configuration
Lower bound
Mean ATE
Upper bound


Configurations' Performance on Sequence 07
ATE Value
0.52
0.5
0.48
0.46
0.44
0.42
0.4
D
Gn21
Gr80
Gn51
Gn68
Gr223
HB5
Gr192
Gr224
Gr235
Configuration
Lower bound
Mean ATE
Upper bound


Configurations' Performance on Sequence 08
ATE Value
3.55
3.45
3.35
3.25
3.15
3.05
2.95
D
Gn21
Gr80
Gn51
Gn68
Gr223
HB5
Gr192
Gr224
Gr235
Configuration
Lower bound
Mean ATE
Upper bound


Configurations' Performance on Sequence 09
4.4
3.9
3.4
2.9
2.4
1.9
1.4
ATE Value
D
Gn21
Gr80
Gn51
Gn68
Gn223
HB5
Gn192
Gn224
Gn235
Configuration
Lower bound
Mean ATE
Upper bound


Configurations' Performace on Sequence 10
1.2
1.1
1
0.9
0.8
0.7
0.6
ATE Value
D
Gr21
Gn80
Gr51
Gr68
Gr223
HB5
Gr192
Gr224
Gr235
Configuration
Lower bound
Mean ATE
Upper bound