AI-Guided Optimization for 3D Gaussian Splatting on Drone Images

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science (MSc)

PARIS-LODRON UNIVERSITY SALZBURG (PLUS)

Faculty of Digital and Analytical Sciences

Department of Geoinformatics

and

PALACKÝ UNIVERSITY OLOMOUC (UPOL)

Faculty of Science

Department of Geoinformatics

PLUS supervisor: Getachew Workineh Gella, PHD

UPOL supervisor: Assoc. Prof. RNDr. Stanislav Popelka, PHD

submitted by

Upendra Oli

Salzburg & Olomouc, July, 2026

With the support of Erasmus+ programme of the European Union. This Master's Thesis has been developed in the framework of the Erasmus Mundus Joint Master Degree (EMJMD) "Copernicus Master in Digital Earth", jointly coordinated by Paris-Lodron University Salzburg, Department of Geoinformatics, Austria together with University of South Brittany, Computer Science Department, France and Palacký University Olomouc, Department of Geoinformatics, Czech Republic.

ANOTATION

This thesis focuses on learning from previous 3D Gaussian Splatting (3DGS) reconstruction projects and using that knowledge to recommend hyperparameters for new drone-image projects. The main goal is to reduce manual trial-and-error in 3DGS hyperparameter selection and to make the reconstruction workflow more practical for repeated drone-based projects.

In this study, each drone project is first reconstructed using default 3DGS settings. After that, controlled changes are applied to selected hyperparameters, and the results are compared with the default reconstruction. The comparison is based on selected image-quality metrics: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS). These results are then further used to prepare training data for learning models.

The proposed workflow uses project descriptors, tested multipliers applied to default hyperparameters, and reconstruction-quality results to recommend a suitable combination of multiplier values for three parameter groups defined in this thesis: geometry-related, appearance-related, and densification-related parameters. The recommended values are used as learned starting settings for new drone-image projects. The results show that information from earlier reconstruction projects can support better hyperparameter selection for future projects. This can save time and effort by reducing repeated manual trial-and-error in drone-based 3D reconstruction.

KEYWORDS

3D reconstruction; UAV photogrammetry; hyperparameter recommendation; project descriptors; supervised learning; Ridge Regression; multilayer perceptron; reconstruction quality;

Number of pages: 66

Number of appendixes: 3

This thesis has been composed by *Upendra Oli* for the Erasmus Mundus Joint Master's Degree Program in Copernicus Master in Digital Earth for the academic years 2024/2025 and 2025/2026 at the Department of Geoinformatics, Faculty of Natural Sciences, Paris Lodron University Salzburg, and Department of Geoinformatics, Faculty of Science, Palacký University Olomouc.

Hereby, I declare that this piece of work is entirely my own, the references cited have been acknowledged, and the thesis has not been previously submitted for the fulfilment of the higher degree.

I also acknowledge the use of GPT-5.6 for language polishing, grammar correction, and improving the clarity of this thesis. During platform development, GPT-5.5 through Codex and Claude Sonnet 4.6 through GitHub Copilot in Visual Studio Code were used to support code suggestions, debugging, and documentation. All AI-assisted text and code were reviewed, tested, and revised. The research design, methodological decisions, data processing, analysis, interpretation, platform testing, and final academic responsibility remain entirely my own.



Upendra Oli,
Olomouc, July 2026

Acknowledgment

I would like to thank my supervisors, Dr. Stanislav Popelka and Dr. Getachew Workineh Gella, for their support throughout this thesis. Our discussions helped me refine the research focus, solve methodological questions, and improve the presentation of the results. Their feedback was especially helpful during the development and evaluation of the 3DGS workflow. I also enjoyed the discussions that helped me narrow down the scope of the thesis and focus on one specific problem rather than several broad topics.

I am also very grateful to Department of Geoinformatics at Paris Lodron University Salzburg and the Department of Geoinformatics at Palacký University Olomouc. The academic support, facilities, and resources available at both universities helped me carry out this research. I also appreciate everyone who shared datasets, provided technical advice, discussed research problems with me, or supported the work at different stages.

Finally, I would like to thank my family and friends for their support and encouragement to complete this work. Their support helped me stay focused throughout my studies and during the final stages of completing this thesis.

Palacký University Olomouc

Faculty of Science

Academic year: 2025/2026

ASSIGNMENT OF DIPLOMA THESIS

(project, art work, art performance)

Name and surname:	**Upendra OLI**
Personal number:	**R241018**
Study programme:	**N0532A330010 Geoinformatics and Cartography**
Work topic:	**AI-Guided Optimization for 3D Gaussian Splatting on Drone Images**
Assigning department:	**Department of Geoinformatics**

Theses guidelines

The aim of this thesis is to design and evaluate an AI-guided workflow that accelerates 3D Gaussian Splatting (GS) on drone imagery, making it faster, more automated, and more reliable for urban (or semi-urban) scene reconstruction and interactive visualization. The work will focus on learning from previously reconstructed scenes to recommend suitable GS training and processing parameters for new areas, reducing manual trial-and-error while maintaining photorealistic rendering quality and real-time performance.

The input will consist of multi-view drone imagery captured over 2–4 selected sites in the Moravia region (or comparable freely available datasets if needed), using different flight strategies.

A baseline workflow will be established using an existing open-source GS implementation (e.g., gsplat, LiteGS, or a comparable framework), including systematic parameter logging and evaluation of convergence time, final visual quality, and rendering performance. The learning-enhanced pipeline will be evaluated against the baseline across multiple test scenes, focusing on training efficiency, reconstruction quality, robustness, and user experience in the interactive viewer.

The thesis will result in (1) a functional prototype that integrates GS reconstruction with an AI-supported parameter recommendation module, (2) an interactive web visualization presenting the reconstructed urban scenes, and (3) a documented methodology including evaluation metrics and recommendations for practical drone-based GS deployments. The work will also briefly discuss the potential for enriching future applications with Copernicus satellite data.

The student will create a website about the thesis following the rules available on the department's website and a poster about the diploma thesis in A2 format. The student will submit the entire text (text, attachments, poster, outputs, input, and output data) in digital form on a storage medium and the text of the thesis in two bound copies to the secretary of the department.

Extent of work report:	**max. 50 pages**
Extent of graphics content:	**as needed**
Form processing of diploma thesis:	**electronic**
Language of elaboration:	**English**

Recommended resources:

Fang, X., Zhang, Y., Tan, H., Liu, C., & Yang, X. (2025). Performance evaluation and optimization of 3D Gaussian Splatting in indoor scene generation and rendering. ISPRS International Journal of Geo-Information, 14(1), 21. https://doi.org/10.3390/ijgi14010021

He, Y., Wang, Y., & Yang, X. (2024). MetaGS: A meta-learned Gaussian-Phong model for out-of-distribution 3D scene relighting. arXiv preprint arXiv:2405.20791. https://doi.org/10.48550/arXiv.2405.20791

Jäger, M., Hillemann, M., & Jutzi, B. (2025). FeatureGS: Eigenvalue-feature optimization in 3D Gaussian Splatting for geometrically accurate and artifact-reduced reconstruction. ISPRS Open Journal of Photogrammetry and Remote Sensing, 17, 100100. https://doi.org/10.1016/j.ophoto.2025.100100

Jung, J., Han, J., An, H., Kang, J., Park, S., & Kim, S. (2024). Relaxing Accurate Initialization Constraint for 3D Gaussian Splatting. arXiv preprint arXiv:2403.09413. https://doi.org/10.48550/arXiv.2403.09413
Kerbl, B., Kopanas, G., Leimkühler, T., & Drettakis, G. (2023). 3D Gaussian Splatting for real-time radiance field rendering. arXiv preprint arXiv:2308.04079. https://doi.org/10.48550/arXiv.2308.04079
Li, Z., Zhan, H., Li, C., Yan, Q., & Xu, Y. (2025). RLGS: Reinforcement Learning-Based Adaptive Hyperparameter Tuning for Gaussian Splatting. arXiv preprint arXiv:2508.04078. https://doi.org/10.48550/arXiv.2508.04078
Liu, Y.-C., Höllein, L., Nießner, M., & Dai, A. (2025). QuickSplat: Fast 3D Surface Reconstruction via Learned Gaussian Initialization. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV 2025). https://doi.org/10.48550/arXiv.2505.05591
Pateux, S., Gendrin, M., Morin, L., Ladune, T., & Jiang, X. (2025). BOGausS: Better Optimized Gaussian Splatting. arXiv preprint arXiv:2504.01844. https://doi.org/10.48550/arXiv.2504.01844

Supervisors of diploma thesis: **doc. RNDr. Stanislav Popelka, Ph.D.**
Department of Geoinformatics

Date of assignment of diploma thesis: **October 1, 2025**
Submission deadline of diploma thesis: **July 30, 2026**

LS.

prof. RNDr. Martin Kubala, Ph.D.
Dean

prof. RNDr. Vilém Pechanec, Ph.D.
Head of Department

Olomouc November 3, 2025

Content

LIST OF ABBREVIATIONS ..... A
Introduction ..... B
1 objectives 1
2 State of the art 2
2.1 3D Reconstruction in Traditional Photogrammetry 2
2.2 Neural Scene Representations 2
2.3 3D Gaussian Splatting 3
2.4 Recent Advances in Gaussian Splatting 4
2.5 Hyperparameter Optimization Methods 6
2.6 Learning from Previous Hyperparameter Experiments 7
2.7 Ridge Regression and Neural Predictive Models 8
2.8 Project Descriptors for Drone Reconstruction 9
2.9 Research Gap 9
3 Methodology 11
3.1 Used Methods and Processing Procedure 11
3.2 Used Data 12
3.3 Used Software and Hardware 13
4 Learning-Based Hyperparameter Selection 15
4.1 Project Descriptor Extraction and SfM Processing 15
4.1.1 Image Resizing and SfM Processing 16
4.1.2 Project Descriptor Extraction 16
4.1.3 Descriptor Standardization 18
4.2 Training Data Preparation 18
4.2.1 Hyperparameter Selection, Grouping, and Bounds 19
4.2.2 Baseline Run 21
4.2.3 Exploration Runs 22
4.2.4 Quality Evaluation 25
4.2.5 Training Rows 27
4.3 Scoring-Model Training 28
4.3.1 Shared Model Input and Learning Target 28
4.3.2 Ridge Regression Scoring Model 29
4.3.3 MLP Scoring Model 31
4.4 Model Testing and Evaluation 32
5 Platform Development 35
5.1 Platform Overview and Project-Based Processing 35
5.2 Research Workflow 37
5.2.1 Project Descriptor Extraction and SfM Processing 37
5.2.2 Training Data Praparation 37
5.2.3 Scoring Model Training 39
5.2.4 Model Testing and Evaluation 39
5.3 Availability and Reuse 41
6 Results 42
6.1 Summary of Results 42
6.2 Offline Data Preparation and Training Data 44
6.3 Scoring Model Training 46

6.4 Evaluation on Test Projects..... 48
6.4.1 Ridge Regression Results..... 48
6.4.2 MLP Results.....53
6.4.3 Supporting Model Comparisons..... 54
6.4.4 Ridge Results at the 7,000-Iteration Checkpoint.....54
6.5 Outputs.....55
6.5.1 3DGS Processing and Recommendation Platform..... 55
6.5.2 Interactive Results Visualization Platform..... 56
6.5.3 Thesis Website and Poster.....56
7 DISCUSSION..... 57
7.1 Interpretation of the Model Results.....57
7.2 Quality, Processing Cost and Run Stability.....59
7.3 Limitations.....61
7.4 Alternative Approaches and Future Development.....61
7.5 Potential Use of Copernicus Data in 3DGS.....62
8 CONCLUSION..... 65
APPENDIXES..... 73
Appendix 1 Datasets.....73
Appendix 2 Coefficients of the final Ridge Regression scoring model..... 78
Appendix 3 Model Results..... 80
Appendix 3A Ridge Regression quality results and selected multiplier combinations.....80
Appendix 3B Ridge Regression Gaussian-count and runtime results.....81
Appendix 3C MLP quality results and selected multiplier combinations.....82
Appendix 3D MLP Gaussian-count and runtime results.....83
Appendix 3E Checkpoint Comparison OF Baseline And Ridge Model-Selected Runs..... 84
Appendix 3F Supporting Models Trained With Hard-Cap Data..... 85
Appendix 3G Repeatability Check Using the Same Multiplier Combinations..... 85

REFERENCES AND INFORMATION SOURCES

ATTACHMENTS..... 86

LIST OF ABBREVIATIONS

Abbreviation	Meaning
3DGS	3D Gaussian Splatting
AI	Artificial Intelligence
API	Application Programming Interface
CUDA	Compute Unified Device Architecture
ESA	European Space Agency
EXIF	Exchangeable Image File Format
GPU	Graphics Processing Unit
GSD	Ground Sampling Distance
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
LASSO	Least Absolute Shrinkage and Selection Operator
LPIPS	Learned Perceptual Image Patch Similarity
MLP	Multilayer Perceptron
MSE	Mean Squared Error
NeRF	Neural Radiance Fields
PSNR	Peak Signal-to-Noise Ratio
ReLU	Rectified Linear Unit
RLGS	Reinforcement Learning-based Gaussian Splatting
RMSE	Root Mean Squared Error
SfM	Structure-from-Motion
SSIM	Structural Similarity Index Measure
UAV	Unmanned Aerial Vehicle
VRAM	Video Random Access Memory

INTRODUCTION

Three-dimensional (3D) reconstruction using drone images is widely used in digital mapping, heritage recording, infrastructure monitoring, and geovisualization. A drone can capture the same area from different viewpoints. It can provide the detailed information about buildings, terrain, vegetation, and other objects. This makes the drone images suitable for creating realistic 3D representations of different environments, especially when the scene needs to be inspected or visualized later (Nex & Remondino, 2014).

Recent developments in neural rendering have created new ways to represent scenes and view them from different camera positions. Mildenhall et al. (2020) showed that Neural Radiance Fields can produce high-quality novel views from images with known camera poses. However, this method needed many network evaluations along camera rays to render the scene. It made the rendering process computationally expensive. Later, Kerbl et al. (2023) introduced 3D Gaussian Splatting (3DGS), which represents the scene using three-dimensional Gaussian primitives and supports efficient rendering after the training. It requires camera poses and sparse scene data before the training. Those required data are commonly prepared using Structure-from-Motion pipelines such as COLMAP (Schönberger & Frahm, 2016).

3DGS can produce high-quality visual reconstructions, but the training result depends on different hyperparameter settings. These settings control how the positions and shapes of the Gaussians are updated, how colour and opacity are learned, and how Gaussians are added or removed during densification. Default values are useful as a starting point. However, the drone projects can differ in image resolution, viewing angle, overlap, texture, vegetation, terrain, and flight pattern. Because of these differences, the same default hyperparameter settings may not work well for all projects. Testing different hyperparameter settings for every new project would require several reconstruction runs and additional processing time. But the previous projects may already contain useful information about project conditions, tested settings, training behaviour, and reconstruction results.

This thesis addresses this problem by developing an AI-guided workflow for recommending 3DGS hyperparameters for drone-image projects. The workflow recommends the bounded multiplier values for a new project using results from previous reconstruction projects . The selected values are then tested in a new 3DGS run and compared with a baseline run that uses the default settings. Project descriptors are calculated from image properties, acquisition information, scene-related indicators, and reconstruction outputs. These descriptors are combined with the results of offline exploration runs to train Ridge Regression and MLP scoring models. The models estimate the expected relative quality of different multiplier combinations. The aim is to reduce repeated manual trial-and-error and examine whether the recommended settings can improve or maintain reconstruction quality compared with the defaults. The thesis also briefly discusses how Copernicus data could complement future 3DGS reconstruction and visualization workflows.

1 OBJECTIVES

The aim of this thesis was to design and evaluate an AI-guided workflow for 3DGS on drone imagery. The workflow focused on learning from previously reconstructed scenes to recommend suitable Gaussian Splatting training hyperparameters for new drone-image projects. The main purpose was to reduce manual trial-and-error in hyperparameter selection and to examine whether recommendations learned from previous projects can improve or maintain reconstruction quality compared with default settings.

To achieve the main aim of the thesis, the following specific objectives were defined:

- To create a baseline 3DGS workflow for drone-image reconstruction using default parameter settings.
- To prepare training data for hyperparameter recommendation by using project descriptors, baseline runs, and controlled hyperparameter exploration runs.
- To develop a learning-based approach for recommending suitable hyperparameter multipliers for new drone-image projects.
- To evaluate whether the recommended hyperparameters improve or maintain reconstruction quality compared with default 3DGS settings.
- To implement the complete workflow in an open-source platform for project preparation, 3DGS reconstruction, training-data generation, model training, recommendation, and result comparison.

The theoretical part of the thesis includes 3DGS training, hyperparameter optimization, and learning-based recommendation. The practical part contains the project preparation, baseline and exploration runs, training-data creation, model development and evaluation, and implementation of the supporting platform.

The results of this thesis support learning-guided hyperparameter selection in drone-based 3DGS workflows. The proposed workflow uses information from previous reconstruction projects rather than manually testing many settings for every new project. It provides a learned starting point for new reconstruction projects, but the final results still depend on the project conditions and the available training data.

The training data prepared in this thesis links each drone project with its project descriptors, tested hyperparameter settings, and quality results. It can support future studies on hyperparameter tuning and reconstruction quality in 3DGS. The open-source platform developed in this thesis can also support both users and researchers. General users can create projects, generate point clouds and 3DGS reconstructions, test recommended settings, and compare the outputs visually. Researchers can add new project runs, expand the training data, train and test scoring models across multiple projects, and compare parameter settings, quality values, and visual results. The thesis also briefly discusses how the Copernicus data could support and complement the drone-based Gaussian Splatting workflows.

2 State of the art

This chapter discusses the research related to AI-guided hyperparameter recommendation for 3D Gaussian Splatting on drone images. It starts with neural scene representations to explain the development of modern image-based reconstruction and novel view synthesis. Then discusses about the 3D Gaussian Splatting as the main reconstruction method used in this thesis. Recent developments in Gaussian Splatting were also reviewed and discussed in terms of efficiency, geometry, compactness, aerial reconstruction, and adaptive optimization. The final parts of the chapter cover hyperparameter optimization, learning from previous 3DGS runs, predictive models, and project descriptors before talking about the research gap addressed by this thesis.

2.1 3D Reconstruction in Traditional Photogrammetry

Traditional UAV photogrammetry is an established approach for 3D reconstruction from drone images. It is commonly used for mapping, inspection, monitoring, and 3D modelling. The outputs include digital surface models, digital terrain models, orthoimages, vector information, and textured 3D models (Nex & Remondino, 2014). Structure-from-Motion (SfM) is generally used in image-based workflows to estimate camera positions, camera orientations, and sparse scene structure, which are important inputs for later dense reconstruction and 3D modelling (Schönberger & Frahm, 2016). These outputs from traditional photogrammetry are important when the main requirement is to produce a measurable spatial product that can be used for mapping, engineering work, or further GIS-based analysis.

However, the traditional photogrammetric reconstruction is mainly based on geometric reconstruction and image-based texturing. It is not designed to learn view-dependent appearance or generate new views in the same way as recent neural rendering methods. This difference provides the background for the neural scene representations discussed in the next section.

2.2 Neural Scene Representations

Neural scene representations became important in novel-view synthesis because they allow a scene to be represented by learnable functions rather than only by meshes, voxels, or point clouds. One of the most influential methods in this area is Neural Radiance Fields, commonly known as NeRF. Mildenhall et al. (2020) introduced NeRF as a method for synthesizing new views of a scene from posed input images. The method represents a scene as a continuous function that maps a 3D position and viewing direction to volume density and view-dependent colour. Images are rendered using differentiable volume rendering, and the model is optimized by comparing rendered 2D views with the input images. NeRF demonstrated that high-quality novel views can be generated from images with known camera poses. However, the original method required many neural network evaluations during rendering, which made rendering slow in the original implementation (Mildenhall et al., 2020). This limitation motivated later research on faster and more explicit scene representations.

Several later methods improved the efficiency of neural rendering. Plenoxels showed that radiance fields could be optimized using a sparse voxel grid without a neural network, which reduced training time compared with the original NeRF-style representation (Fridovich-Keil et al., 2022). Instant-NGP introduced a multiresolution hash encoding that allowed neural graphics primitives to be trained much faster with a smaller neural network (Müller et al., 2022). These developments are relevant because they show a wider movement from purely implicit neural representations toward more efficient explicit or hybrid representations.

For drone-based reconstruction and geovisualization, interactive rendering is important. Drone imagery is often used to document urban areas, infrastructure, heritage objects, terrain, and vegetation. In these applications, reconstruction quality is important, but users also need to inspect reconstructed scenes from different viewpoints. Therefore, methods that combine visual quality with efficient rendering are especially useful for practical spatial visualization.

2.3 3D Gaussian Splatting

3D Gaussian Splatting (3DGS) was introduced by Kerbl et al. (2023) as a method for real-time radiance field rendering. Instead of representing a scene only through a neural network, the method uses a set of three-dimensional anisotropic Gaussian primitives. Each Gaussian has learnable properties such as position, opacity, covariance, and colour represented using spherical harmonics. The method starts from sparse points obtained from SfM and optimizes the Gaussian representation using posed input images.

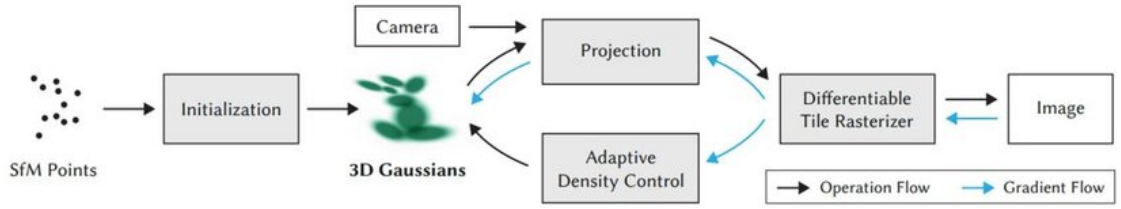


Figure 1 Overview of the 3DGS pipeline (Source: Kerbl et al., 2023).

As shown in Figure 1, 3DGS begins from sparse SfM points, which are used to initialize a set of 3D Gaussians (Kerbl et al., 2023). These Gaussians form the scene representation and are optimized from the input images. During training, the Gaussians are projected into the image plane and rendered with a differentiable tile-based rasterizer. At the same time, adaptive density control refines the representation by adjusting the set of Gaussians as optimization proceeds. Kerbl et al. (2023) described the method through three main components: an explicit 3D Gaussian representation, interleaved optimization with adaptive density control, and a visibility-aware rasterization procedure for efficient rendering. Together, these components allow 3DGS to produce high-quality novel views and render the trained scene in real time.

The way a scene is represented in 3DGS is important because it changes how reconstruction and rendering are handled. NeRF renders an image through repeated neural network evaluations along camera rays. In 3DGS, the optimized scene is stored as a collection of Gaussian primitives that can be rasterized efficiently. This makes the method suitable for the interactive visualization, including web-based inspection of reconstructed scenes.

Before starting the reconstruction process, 3DGS requires camera poses and sparse point clouds. These inputs are commonly prepared using an SfM workflow like COLMAP. Schönberger and Frahm (2016) describe the main COLMAP steps, including feature extraction and matching, geometric verification, incremental image registration, triangulation, and bundle adjustment. The resulting camera poses and sparse points provide the inputs needed to initialize the original 3DGS method (Kerbl et al., 2023).

Recent advances in software are making it easier to conduct a series of 3DGS experiments. Ye et al. (2025) developed gsplat-an open-source 3DGS library-with Python/PyTorch as a platform for implementing bindings as well as optimized CUDA kernels. The library is useful for this thesis because it implements the 3DGS functionality. It also includes the performance improvements and provides a modular API for experimentation. Ye et al. (2025) also compared *gsplat* with the original 3DGS implementation. They reported similar rendering quality with reduced memory use and training time. This makes *gsplat* suitable for repeated reconstruction experiments, including the baseline evaluation and hyperparameter exploration.

At the same time, 3DGS contains many hyperparameters that can influence the reconstruction behaviour. These include learning rates for Gaussian attributes, opacity-related parameters, loss-balancing terms, and densification thresholds. The original method provides default optimization settings, but later work on adaptive tuning indicates that 3DGS training can be affected by hyperparameter choices such as learning rates and densification thresholds (Kerbl et al., 2023; Li et al., 2025). Li et al. (2025) also demonstrated that the reinforcement learning-based adaptive hyperparameter tuning can improve rendering quality across different datasets. This makes 3DGS a suitable target for learning-based hyperparameter recommendation.

2.4 Recent Advances in Gaussian Splatting

After the introduction of 3DGS, many studies have tried to improve different parts of the method. Some of them have focused on faster training and model compression, while others have explored geometric accuracy, learned initialization, aliasing control, aerial reconstruction, and adaptive optimization.

Some studies have tried to make 3DGS models smaller and more efficient. LightGaussian reduces the size of the Gaussian representation through pruning, knowledge distillation, and quantization (Fan et al., 2024). This is useful because a 3DGS scene can contain millions of Gaussians, which increases storage requirements and may make rendering or deployment more difficult. BOGausS follows a similar direction by improving the optimization process to produce lighter models while keeping the rendering quality stable (Pateux et al., 2025). These studies show that the number

of Gaussians and how they are represented have a strong effect on the efficiency and practical use of 3DGS.

Other studies have focused on improving the geometric quality of 3DGS models. FeatureGS introduced geometric loss terms based on 3D shape features to improve surface accuracy and reduce floater artifacts (Jäger et al., 2025). This is especially relevant for geospatial and drone-based reconstruction, where a visually realistic result may still be insufficient if the shape of the scene is not represented well. Better geometry is particularly important for buildings, terrain, and heritage objects, where surface quality and spatial consistency matter.

Recent work has also explored learned initialization. QuickSplat uses learned Gaussian initialization and densification to speed up surface reconstruction (Liu et al., 2025). Although it was developed for a different reconstruction setting, it shows how knowledge learned from earlier data can support later Gaussian-based reconstruction. This idea is also relevant to this thesis, which uses information from previous reconstruction projects to guide new ones.

Scale changes and aliasing are another challenge in Gaussian Splatting. Mip-Splatting reduces these artifacts by applying filtering and smoothing when the viewing distance or image scale changes (Yu et al., 2024). This is relevant for drone imagery because images are often captured from different heights, angles, and resolutions. Such differences can affect how small details appear in the reconstructed and rendered scene.

The effect of parameter settings on 3DGS has also been examined. Fang et al. (2025) tested settings such as image resolution, the spatial-position gradient threshold, the scaling learning rate, and a density-level parameter in an indoor scene. Their results showed that changing these settings affected PSNR, SSIM, and LPIPS. This indicates that 3DGS quality can depend on the selected parameter values. However, their work used manual parameter testing in an indoor setting, whereas this thesis recommends multiplier values for drone-image projects using information learned from previous reconstructions.

Adaptive hyperparameter tuning has also been studied in Gaussian Splatting. Li et al. (2025) introduced Reinforcement Learning-based Gaussian Splatting (RLGS), which changes selected hyperparameters while 3DGS training is running. The method treats tuning as a sequence of decisions and updates the settings according to the current training behaviour. This study is relevant because it shows that learning-based control of Gaussian Splatting hyperparameters is an active research direction. However, it differs from the present thesis. RLGS performs online adaptation during training, while this thesis learns from offline exploration data generated using different past projects and recommends multiplier values before training begins.

The reviewed Gaussian Splatting studies mainly improve the reconstruction method itself. They change the representation, optimization process, compression strategy, geometry constraints, initialization, or aerial-scene handling. Less attention has been given to a lightweight workflow that reuses previous drone reconstruction projects to predict suitable hyperparameter adjustments for new projects before training starts. Addressing this limitation requires both a practical exploration strategy and a method

for learning from previous experiments. Hyperparameter optimization methods are therefore reviewed next.

2.5 Hyperparameter Optimization Methods

Hyperparameter optimization is a common problem in machine learning because model performance often depends on settings chosen before the training. These settings are not learned in the same way as model parameters, but they can strongly influence training behaviour and final performance. Common approaches include manual tuning, grid search, random search, Bayesian optimization, and adaptive resource-allocation methods (Bischl et al., 2023). Grid search evaluates predefined combinations of hyperparameter values. It is simple and reproducible, but the number of required experiments increases quickly when more parameters or values are added. This makes grid search difficult to apply in this thesis because each Gaussian Splatting evaluation requires a computationally heavy reconstruction run, making large numbers of samples time-consuming to produce.

Random search samples configurations from predefined search spaces or probability distributions. Bergstra and Bengio (2012) showed that random search can be more efficient than grid search when only some hyperparameters strongly affect performance. Their experiments also used log-scaled distributions for some hyperparameters, which supports sampling in log space when multiplicative changes are more meaningful than additive changes. Random search is therefore useful as a simple approach for hyperparameter exploration. This idea is relevant to this thesis because the multiplier exploration also required sampling from a bounded search space rather than testing all possible combinations.

Bayesian optimization uses previous evaluations to guide the selection of new configurations. Snoek et al. (2012) applied this method to machine-learning hyperparameters using a probabilistic model of the objective function and an acquisition function to select new trials. It is useful when each evaluation is expensive. However, it is usually carried out as a sequential search for one task or dataset. This means that the several trials may still be needed on the same project before a suitable configuration is identified.

Hyperband is an adaptive method for allocating resources during hyperparameter search (Li et al., 2018). It gives more resources to promising configurations and stops weaker ones at an early stage. This can reduce the cost of testing many configurations. However, Hyperband is mainly designed as a search method and does not directly learn a reusable relationship between project descriptors and hyperparameter multipliers. It was also less suitable for this thesis because both better and worse performing exploration runs were needed as training observations.

Latin Hypercube Sampling is another useful strategy when only a limited number of experiments can be run. It provides a practical way to sample the full range without testing every possible combination. Each variable range is divided into intervals and values are sampled across these intervals so that all parts of the range are represented. McKay et al. (1979) introduced this approach for computer experiments where a complete exploration can be too expensive. The same principle can be used to test the

hyperparameters and produce a small but well-distributed set of configurations within fixed bounds.

These methods provide different ways to explore hyperparameter values. Latin Hypercube Sampling was selected for the exploration stage of this thesis because it helps to spread the limited runs across the bounded search space. This was important because the multipliers were applied together in each run rather than one at a time. This helped to examine their combined effects on the 3DGS reconstruction. Chapter 4 explains the detailed sampling procedure, including the bounded log-transformed multiplier space.

2.6 Learning from Previous Hyperparameter Experiments

Learning-based hyperparameter optimization aims to reuse information from previous optimization experience. This idea is related to transfer learning for hyperparameter optimization. Feurer et al. (2015) used past performance on similar datasets to support automated algorithm selection, preprocessing selection, and hyperparameter configuration in auto-sklearn. This shows that information from previous tasks can help guide the search for suitable configurations on a new dataset. This study is relevant because they treat previous optimization experience as useful information for future tasks. However, they are general hyperparameter-optimization studies and are not designed specifically for Gaussian Splatting. They do not directly address the structure of 3DGS, where geometry, appearance, and densification parameters affect different parts of the reconstruction process.

Existing work such as RLGS adapts selected Gaussian Splatting hyperparameters during training (Li et al., 2025). Other learned approaches focus on different parts of the Gaussian Splatting workflow, such as learned initialization and densification in QuickSplat (Liu et al., 2025), or optimization behaviour and model compactness in BOGausS (Pateux et al., 2025). These studies are useful for showing that learning-based control of Gaussian Splatting is an active research direction, but they do not directly learn from previous drone reconstruction projects to recommend bounded hyperparameter multipliers before training begins.

In drone-based Gaussian Splatting, previous reconstruction projects may contain information about how scene properties and image conditions affect training behaviour. For example, image quality, view geometry, overlap, vegetation, texture, and scene coverage may influence which parameter settings are suitable. Therefore, the final approach in this thesis is closer to offline supervised recommendation than to online reinforcement learning (Li et al., 2025). The system does not update a policy during deployment. Instead, offline exploration is first used to generate observations linking project descriptors, tested multipliers, and reconstruction outcomes. These observations are then used to train predictive models.

This makes the approach practical for drone reconstruction, where repeated experiments on every new project can be costly. In applied drone reconstruction, repeated trial-and-error on every new project can be expensive. A model that uses descriptors from previous projects may reduce the need for manual tuning, even if it does not fully replace detailed optimization.

2.7 Ridge Regression and Neural Predictive Models

Ridge Regression was introduced by Hoerl and Kennard (1970) as a linear regression method for situations where the input variables are correlated. It adds a penalty that shrinks coefficient values towards zero and reduces large or unstable estimates (Schreiber-Gregory, 2018). This is relevant to drone reconstruction because some project descriptors, such as overlap, coverage spread, vegetation, and terrain roughness, may be related to each other.

Regularized linear models are useful when the prediction model needs to remain simple, stable, and interpretable. Ridge Regression keeps all input variables in the model but reduces very large or unstable coefficient values. This can be useful when only a limited number of independent training projects are available. Chaibub Neto et al. (2014) compared Ridge Regression with LASSO and Elastic Net with limited samples. Ridge performed better than LASSO and remained competitive with Elastic Net, especially when the sample size was small and the outcome depended on several input variables. LASSO uses L1 regularization and can reduce some coefficients to zero, which can remove those variables from the model (Tibshirani, 1996). Elastic Net combines L1 and L2 penalties to balance variable selection and coefficient stability (Zou & Hastie, 2005). These methods are useful in many prediction problems. However, LASSO may remove some variables completely, while Elastic Net introduces an additional balance between L1 and L2 regularization.

Other regression models can learn nonlinear relationships between inputs and outputs. Gaussian Process Regression is a kernel-based method that gives both a prediction and an estimate of how uncertain that prediction is (Rasmussen & Williams, 2006). It is often used for hyperparameter optimization when each experiment is costly and only a small number of settings can be tested (Snoek et al., 2012). Support Vector Regression can also learn nonlinear patterns by using kernel functions and margin-based optimization (Smola & Schölkopf, 2004). Random Forest Regression combines many decision trees to capture complex patterns and relationships between variables (Breiman, 2001). These models can be helpful when the relationship between the inputs and the output is not simple. However, they often need more tuning and are usually harder to explain than linear models.

Multilayer perceptrons (MLPs) provide another way to model nonlinear relationships between project descriptors, multiplier combinations, and reconstruction quality. They can learn complex patterns through layers of nonlinear transformations and are widely used in machine-learning tasks. However, MLPs require sufficient training data and careful choices of architecture, regularization, and training settings. When the number of independent training projects is small, MLPs may overfit tabular data and generalize poorly. Studies on real-world tabular datasets show that deep-learning models do not always outperform traditional machine-learning methods and that strong traditional baselines remain important for supervised tabular prediction tasks (Borisov et al., 2024).

Therefore, Ridge Regression is suitable when stability and interpretability are important, while a compact MLP can represent possible nonlinear relationships. In both cases, prediction depends on project descriptors that represent the conditions of each drone-image project. The role of these descriptors is discussed in the next section.

2.8 Project Descriptors for Drone Reconstruction

Project descriptors can be used to represent the conditions under which a drone-image reconstruction is performed. In the context of hyperparameter recommendation, such descriptors are useful because they describe image quality, camera geometry, acquisition structure, scene content, and reconstruction difficulty. Their selection is supported by previous work in SfM, Gaussian Splatting, UAV photogrammetry, and image-based 3D reconstruction.

SfM relies on reliable image correspondences, camera geometry, and sufficient view coverage to reconstruct 3D scenes from images. Schönberger and Frahm (2016) describe SfM as a 3D reconstruction pipeline involving image registration, triangulation, and bundle adjustment. Similarly, the original 3DGS method starts from sparse points produced during camera calibration and optimizes the scene representation from input views (Kerbl et al., 2023). For this thesis, these requirements provide a basis for using descriptors such as overlap, camera angle, heading consistency, and coverage spread. The descriptors do not directly determine reconstruction quality. Instead, they describe project conditions that may influence image matching, camera-pose estimation, and later 3DGS training.

UAV photogrammetry provides a wider background for image acquisition and 3D reconstruction from drone images. UAVs are commonly used for mapping and 3D modelling, and the quality of the final products depends partly on the conditions of image acquisition and processing (Nex & Remondino, 2014). For this reason, the descriptor set also includes information about image detail, image quality, and scene content. Ground sampling distance (GSD) describes the spatial detail in the images. Blur risk and texture density describe image conditions, while vegetation, coverage spread, and terrain roughness describe the scene being reconstructed.

These descriptors are not assumed to predict the best 3DGS hyperparameters on their own. Instead, they are used as practical indicators for reconstruction conditions that may influence training behaviour. They are used in this thesis to examine whether information from previous drone projects can support hyperparameter recommendation for new projects. The research gap related to this approach is discussed in the next section.

2.9 Research Gap

The reviewed literature shows that 3DGS has developed rapidly since its introduction in 2023. Recent studies have improved rendering efficiency, compactness, geometric accuracy, initialization, aliasing control, aerial reconstruction, and adaptive tuning. However, most of these studies modify the reconstruction method itself or adapt parameters during a single training process. They do not directly focus on reusing previous drone reconstruction projects to recommend hyperparameter multipliers for future projects.

The hyperparameter-optimization literature provides search methods such as random search, Bayesian optimization, and Hyperband. Related work on transfer learning for hyperparameter optimization also shows that information from previous datasets or

optimization tasks can support new hyperparameter searches. However, these approaches are not designed specifically for drone-based Gaussian Splatting, where project descriptors may relate differently to geometry-related, appearance-related, and densification-related parameter groups.

This thesis addresses this gap by studying ***whether project descriptors from earlier drone-image reconstructions can support hyperparameter recommendations for new projects***. The proposed approach does not replace 3DGS or search for the best settings through many repeated runs. Instead, it tests an offline recommendation workflow that uses information from previous projects to provide a learned starting point and reduce manual trial-and-error.

3 Methodology

This chapter explains the methods, data, software, and processing steps used to develop and evaluate the proposed 3DGS hyperparameter recommendation workflow. The study included drone-image preparation, 3DGS reconstruction, training-data creation, scoring-model training, and evaluation on unseen projects.

3.1 Used Methods and Processing Procedure

The methodology contains four major steps. First, project descriptors and SfM outputs were prepared for the drone-image projects. Second, training data was prepared. Third, the models were trained. Finally, the trained models were tested. The figure below presents this overall workflow.

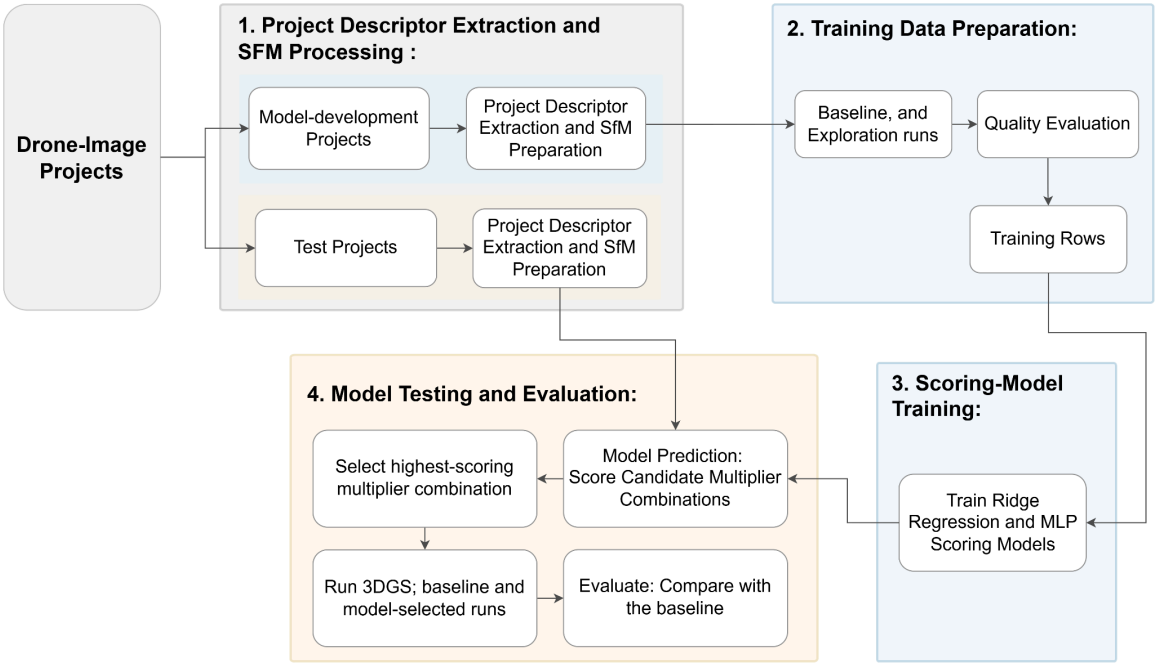


Figure 2 Overview of the proposed 3DGS hyperparameter recommendation methodology.

The workflow started with a collection of drone-image projects. In **Step 1**, the drone-image projects were divided into model-development projects for model training and test projects for testing the trained model. The split was made at the project level so that all data from the same project stayed in the same group. This helped ensure that the final test projects remained unseen during model development.

Both groups followed the same initial preparation process. SfM processing was used to generate the camera poses and sparse point clouds needed by 3DGS. Camera poses describe the position and viewing direction of each camera, while the sparse point cloud provides the initial 3D scene structure needed to start 3DGS training. **Project descriptors** were also prepared from the images, metadata, SfM outputs, and derived scene and acquisition information. These descriptors represented project characteristics

that could affect reconstruction, such as image detail, image overlap, viewing geometry, texture, blur risk, terrain variation, and vegetation.

In **Step 2**, the model-development projects were used to create the training data. First, a **baseline run** was completed for each project using the default hyperparameters. Then, additional **exploration runs** were carried out after applying different multipliers to selected default hyperparameters. Here, a **multiplier** means a value that increases or decreases a default value by multiplying it. Whereas the **multiplier combination** contains the set of multipliers applied together in one exploration run. Then each exploration result was compared with the baseline from the same project. This comparison produced an internal **relative quality score** showing whether the tested combination performed better or worse than the default settings. The score was described as internal because it was not used to present the final result in the evaluation. But it is used as the learning target for the scoring models as an intermediate value. Each usable exploration result formed one **training row** that contained the project descriptors, the tested multiplier combination, and the relative quality score.

In **Step 3**, these training rows were used to train two scoring models. **Ridge Regression** was used as the main model, while a Multilayer Perceptron (**MLP**) was included as a comparison model. Both models learned from the project descriptors, tested multiplier combinations, and their relative quality scores. The models were not trained to directly predict the multiplier values. But they are trained to score multiplier combinations. This was necessary because the best combination was not known for each project. Only the tested combinations and their measured results were available. After training, the models could estimate the expected relative quality of new multiplier combinations for test projects.

In **Step 4**, the trained models were evaluated on the separate test projects. The same descriptor extraction and SfM processing steps were first completed for each project. Then, several **candidate multiplier combinations** were generated within the defined bounds. Each candidate represented one possible set of multiplier values for the selected hyperparameters. The project descriptors and candidate combinations were passed to the scoring model, which predicted a relative quality score for each candidate. The combination with the highest predicted score was selected as the **model-selected multiplier combination**. After selecting the highest predicted score, it was applied to the default 3DGS hyperparameters and tested in a new reconstruction run. Finally, this run was compared with the baseline from the same project to determine whether the reconstruction quality improved, remained similar, or decreased.

Chapter 4 describes these four steps in detail, while Chapter 5 presents the platform developed to support the workflow.

3.2 Used Data

The study used raw drone-image datasets prepared for 3D reconstruction. The projects included oblique and mixed-view image collections acquired under different flight and viewing conditions. These differences were later described using project descriptors related to camera angle, heading consistency, overlap, and coverage spread. The details

of these descriptors is explained in Section 4.1.2, while their distribution across the model-development projects is presented in Section 6.2.

The final dataset contained 58 drone-image projects as listed in **Appendix 1**. The projects differed in scene type, image count, geographic location, flight pattern, data source, and image characteristics. Where image counts were available, the projects ranged from less than 100 images to more than 1,000. Some larger datasets were divided into two to four project cases according to the area they covered. Each divided case represented a different part of the original scene and added further scene-level variation to the dataset. The original image dimensions ranged from 960 × 544 pixels to 9504 × 6336 pixels. These values were recorded as part of the source-dataset description. Before processing, all images were resized so that their longest side did not exceed 1,400 pixels, as explained in Section 4.1.1. This kept the experiments consistent and made processing feasible on the available hardware while retaining sufficient image detail for the comparative 3DGS experiments. The 58 projects were then divided into 46 model-development projects and 12 test projects. The model-development projects were used to prepare the baseline and exploration runs for model training. The test projects were kept separate and used only for the final evaluation. The complete project split is provided in **Appendix 1**.

The datasets came from two main source groups: institutional or privately provided data and publicly available data. The institutional and private projects were provided by the Department of Geoinformatics at Palacký University Olomouc, the Department of Geoinformatics at Paris Lodron University Salzburg, Geovation Nepal, the National Heritage Institute in the Czech Republic, and other academic or professional sources. These datasets were used with permission for academic research and are not redistributed with the thesis. Only derived project descriptors, reconstruction metrics, training logs, and summarized results are reported. Public datasets were added to include a wider range of project conditions. They included photogrammetry and drone-image datasets from Pix4D example projects (PIX4D Documentation, n.d.), Esri sample drone datasets (ESRI, n.d.), UAVID3D (Fernandes et al., 2023), Heritage3DMTL (Shende et al., 2024), and UAV RGB imagery of pistachio trees in Spain (Vélez et al., 2022).

The dataset covered buildings, heritage sites, urban areas, settlements, infrastructure, terrain, vegetation, and mixed scenes. The projects also represented several geographic context, including the Czech Republic, Nepal, Spain, Romania, Lithuania, Canada, and other locations included in the public datasets. This variation allowed the study to examine whether descriptors learned from earlier drone-image projects could help the scoring models evaluate multiplier combinations for unseen projects.

3.3 Used Software and Hardware

The reconstruction workflow used *gsplat v1.5.3*, an open-source 3DGS implementation which is optimized for efficient training and rendering. Input images were preprocessed using *COLMAP 4.0.2* with *CUDA 12.1* support for SfM. COLMAP functions were accessed as pre the through *pycolmap 4.0.2*. This produced the camera poses and intrinsic parameters required for Gaussian Splatting initialization. EXIF metadata was extracted using *piexif 1.1.3*.

Custom Python scripts were used to support the full pipeline, including EXIF extraction and camera parameter handling, descriptor extraction and image analysis, hyperparameter sampling and optimization, model training and training-log parsing, reconstruction evaluation, and quality metric calculation. The scripts are available on GitHub as a part of the platform discussed in Chapter 5.

The learning-based recommendation system used *NumPy* to implement the Ridge Regression scoring model, while *PyTorch 2.5.1+cu121* was used to implement multilayer perceptron models for comparison. *TensorBoard 2.20.0* was used for logging and visualizing training information such as loss curves, number of Gaussians, memory usage, and validation performance. The web platform was developed with *FastAPI 0.135.1 backend and the frontend based on React 18.3.1*, *TypeScript 5.9.3*, *Vite 7.3.0*, *Tailwind CSS 3.4.19*, and *Three.js 0.182.0* .

Experiments were performed on a GPU enabled system with:

- **NVIDIA GeForce RTX 3070 Laptop GPU** (8 GB VRAM) – primary GPU for COLMAP processing, gsplat training and rendering
- **Processor**: AMD Ryzen 9 5900HX (16 cores, ~3.3 GHz)
- **System RAM**: 32 GB

GPU acceleration was essential for gsplat's training efficiency. Runtime measurements were recorded as supporting data but not used as primary learning objectives, since runtime depends on GPU model, VRAM configuration, system load, dataset size, and software optimization levels.

4 Learning-Based Hyperparameter Selection

This chapter describes the detailed implementation of the learning-based hyperparameter selection workflow introduced in Chapter 3. The workflow learns from completed drone-image projects and uses this information to support hyperparameter selection for new projects before 3DGS training begins.

The figure below presents the workflow in four steps: Project Descriptor Extraction and SfM Processing, Prepare Training Data, Train Model, and Test and Evaluate Model.

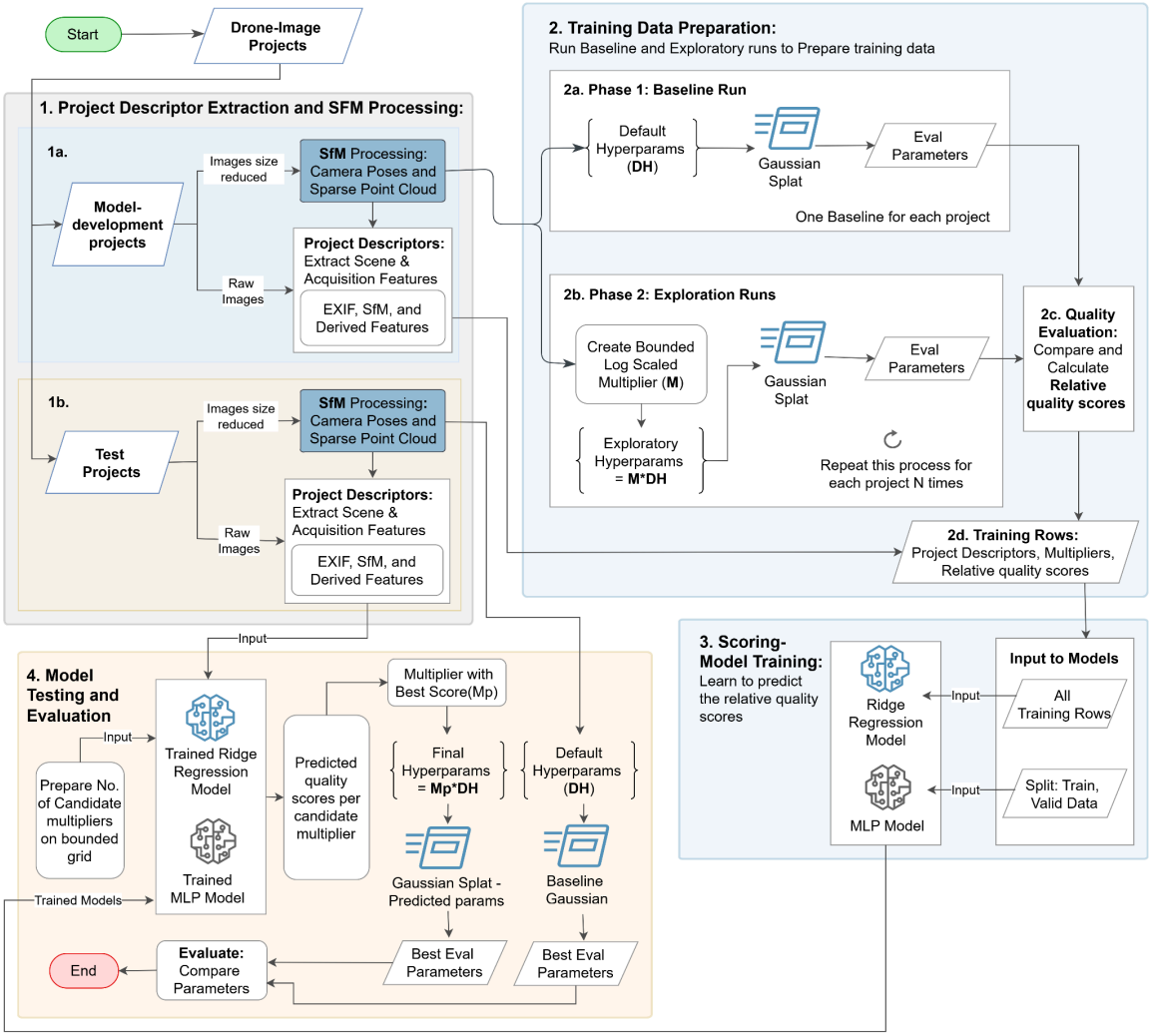


Figure 3 Detailed workflow of the learning-based hyperparameter selection process.

4.1 Project Descriptor Extraction and SfM Processing

This section explains **Step 1 of Figure 3**. It describes how the projects were divided, how the images and SfM outputs were prepared, and how the final project descriptors were calculated and standardized.

The dataset contained 58 drone-image projects. Before model development, 12 projects were separated for final testing. The remaining 46 projects were used as

model-development projects. These images were resized for SfM processing and raw images were used to extract the project descriptors as discussed in sections below.

4.1.1 Image Resizing and SfM Processing

The original drone images were used for metadata extraction and for descriptors that depended on the source images. For SfM processing and 3DGS training, the images were resized so that the longest side did not exceed 1,400 pixels. This limit reduced GPU memory use and processing time across the repeated runs. The same resizing rule was used for the baseline, exploration, and model-selected runs. So, different hyperparameter settings for the same project were compared using same image resolution.

The 1,400-pixel limit was selected as a practical setting rather than the optimal resolution for every project. The official 3DGS reference implementation is designed for input images of about 1–1.6 K pixels and automatically resizes images wider than 1,600 pixels (Graphdeco-Inria, n.d.). So, the selected limit remained within this commonly used range while leaving more GPU memory available for repeated processing on the 8 GB GPU. Fang et al. (2025) tested image resolutions from 0.3 K to 2.9 K and found that resolution affects both reconstruction quality and training time. They selected 0.8 K as a practical resolution for their indoor dataset, but this cannot be treated as the best setting for drone-image projects. In this study, the same 1,400-pixel limit was used for all baseline, exploration, and model-selected runs to keep the comparisons consistent. Nearby limits, such as 1,300 or 1,500 pixels, were not tested because finding the optimal image resolution was outside the scope of the study. Testing several resolutions could have provided more information about the effect of image size on reconstruction quality. However, each additional resolution would have required a new resized image set, SfM processing, a baseline run, and exploration runs for every project. This would have greatly increased the processing workload by multiple times in a study that already required hundreds of 3DGS runs.

After resizing, the images were processed in COLMAP using the SfM workflow. This produced the camera poses, camera parameters, and sparse point cloud needed to initialize 3DGS. These outputs are consistent with the 3DGS input requirements described in Section 2.3 (Schönberger & Frahm, 2016; Kerbl et al., 2023).

The SfM outputs for each project were saved and reused in the baseline, exploration, and model-selected runs. This helped to avoid repeating the same SfM processing and kept the comparison focused on changes caused by the tested hyperparameter combinations.

4.1.2 Project Descriptor Extraction

Project descriptors were used to describe how the drone-image projects were different from one another. Their selection was guided by the reconstruction factors discussed in Section 2.8, including image detail, overlap, viewing geometry, flight regularity, texture, blur, terrain variation, and vegetation.

The descriptor values were obtained from image metadata, GPS and flight information, SfM outputs, and measurements calculated from selected resized images. These image-based measurements included texture, image sharpness, and vegetation cover.

The final Ridge Regression and MLP scoring models used the same set of 10 project descriptors. The project descriptors remained the same for all runs of one project because the images, flight pattern, and scene are the same for one project. The descriptor set was therefore kept compact to avoid using several variables that described nearly the same project characteristic.

Table 1 Final project descriptors, calculation methods, and represented project characteristics

Descriptor	**How it was calculated**	**Value range**	**What it represents**
Median Ground Sampling Distance	Estimated from flight altitude, focal length, sensor width, and processed image width.	Continuous value	Real-world image scale and ground detail.
Estimated Image Overlap	Estimated from the distance between consecutive image positions in relation to flight altitude. Larger spacing produced a lower overlap estimate.	0 to 1	Approximate image overlap and repeated scene coverage.
Image Coverage Spread	Calculated from the geographic spread of the image positions and scaled to a value between 0 and 1.	0 to 1	How widely the images cover the scene.
Camera-Angle Category	Calculated from camera angles obtained through SfM and classified as no usable angle samples, nadir/near nadir, or oblique/mixed.	Categories: 0, 1, 2	Main viewing-angle pattern of the project.
Heading Consistency	Calculated from changes in flight direction between consecutive images. Smaller changes produced a higher consistency value. (Range: 0–1)	0 to 1	Regularity of the flight direction and image-capture pattern.
Texture Density	Estimated from changes in brightness between nearby pixels in selected resized images.	Continuous value	Amount of visible detail that may support image matching and appearance reconstruction.
Blur and Motion Risk	Estimated from image sharpness in selected resized images. Lower sharpness produced a higher blur-risk value. (Range: 0–1)	0 to 1	Risk of blurred or motion-affected images.
Terrain Roughness	Calculated from the height variation of the SfM sparse 3D points. (Range: 0–1)	0 to 1	Surface variation and possible geometric reconstruction difficulty.
Vegetation Complexity	Calculated from vegetation cover and texture density.	Continuous value	Complexity of vegetation appearance and structure.
Vegetation Cover	Estimated from the proportion of green-dominant pixels in selected resized images. (Range: 0–1)	0 to 1	Amount of vegetation visible in the scene.

The descriptors were not assumed to determine the best multiplier combination by themselves. Instead, the scoring models tested whether the descriptors, together with tested multiplier combinations, were related to changes in reconstruction quality.

4.1.3 Descriptor Standardization

Before model training, the 10 project descriptors were standardized.

$$x' = \frac{x - \mu}{\sigma} \quad (1)$$

where x is the original value of descriptor for a project, μ is the mean of that descriptor, σ is its standard deviation, and x' is the standardized value.

The mean and standard deviation were calculated only from the training data. They are then applied to the validation projects for MLP and final test projects for both projects. This means that the descriptor values were different for each validation and test project, but the mean and standard deviation values remain the same. The same training mean and standard deviation were used so that the model received all inputs in the scale it had learned.

Standardization placed descriptors with different units and numerical ranges on a comparable scale. The camera-angle category was stored as a numeric category and standardized with the other descriptors. A minimum standard deviation of 0.01 was used to avoid division by zero. The standardization was important for Ridge Regression because its coefficients and regularization are affected by input scale. The same standardized descriptors were also used by the MLP so that both scoring models received the same project inputs.

The Ridge Regression coefficients were later inspected as model-based indicators of descriptor contribution. They were not treated as evidence that a descriptor directly caused a change in reconstruction quality. The prepared descriptors and SfM outputs from the model-development projects were used for training-data preparation in Section 4.2. The same preparation steps were later applied to the test projects during model testing and evaluation in Section 4.4.

4.2 Training Data Preparation

This section explains **Step 2 of Figure 3**. The model-development projects were used to prepare training data through baseline and exploratory 3DGS runs. Before these runs were started, the hyperparameters, multiplier groups, and exploration bounds were defined. The run results were then evaluated and stored as training rows for the scoring models.

4.2.1 Hyperparameter Selection, Grouping, and Bounds

The first step was to decide which 3DGS hyperparameters should be explored. The implementation includes many training settings, but testing all of them independently would create a very large search space. It would also require more experiments than could be completed with the available projects, hardware, and time. So, the study focused on parameters linked to three main parts of 3DGS training: geometry optimization, appearance optimization, and adaptive densification.

The **geometry** learning rates help to control how quickly the positions, scales, and rotations of the Gaussians are updated. Meanwhile, the **appearance** settings help to control colour features, opacity, and the balance between the image-comparison losses used during training. Similarly, the **densification** settings affect when Gaussians are added, refined, or removed. These parameters may influence reconstruction quality, training behaviour, processing time, and the final number of Gaussians when considered together. Kerbl et al. (2023) describe that the optimization of Gaussian position, scale, rotation, opacity, and colour, together with adaptive densification and pruning. In this thesis, the selected settings were organized into geometry, appearance, and densification groups as a practical design choice. Li et al. (2025) also showed that learning rates and densification-related settings can be adjusted during training and can affect the rendering quality. So, these parameters were considered suitable for controlled exploration.

Grouping the eight adjustable hyperparameters into three multiplier values reduced the exploration space, kept related parameter changes consistent, and made the model recommendations easier to interpret. However, these groups were not tested separately. One multiplier from each group was always applied together as a complete **multiplier combination** in every exploration run.

Table 2 Selected 3DGS hyperparameter groups, default values, and roles

Group	Hyperparameter	Default value	Main role
Geometry	position_lr_init	0.00016	Controls updates to Gaussian positions.
	scaling_lr	0.005	Controls updates to Gaussian scales.
	rotation_lr	0.001	Controls updates to Gaussian rotations.
Appearance	feature_lr	0.0025	Controls colour and appearance learning.
	opacity_lr	0.05	Controls opacity learning.
	lambda_dssim	0.2	Controls the balance between pixel error and structural similarity.
Densification	densify_grad_threshold	0.0002	Controls when Gaussians are created or refined.
	opacity_threshold	0.005	Controls the removal of low-opacity Gaussians.

The default values were taken from the gsplat training configuration used in this study. They are mostly consistent with the original 3DGS settings reported by Kerbl et al.

(2023). One difference is the opacity learning rate. The original implementation used opacity_lr = 0.025, while the gsplat configuration used here had opacity_lr = 0.05.

Multipliers were used instead of adding a fixed amount to the selected hyperparameters. The parameters have very different numerical scales. For example, adding 0.1 to position_lr_init = 0.00016 would change it far more strongly than adding 0.1 to opacity_lr = 0.05. Multipliers preserve proportional changes across parameters. So, the fixed addition would not create the comparable changes.

For each group, the same multiplier was applied to all of its default hyperparameters. For example, the geometry multiplier was applied to the default position, scaling, and rotation learning rates. A multiplier of 1.0 means the default value remained unchanged. A value below 1.0 reduced the hyperparameter, while a value above 1.0 increased it. The same multiplier was applied to all selected hyperparameters within one group.

The multiplier ranges were centered around the default multiplier of 1.0. This allowed both smaller and larger values to be tested while keeping the exploration within controlled limits. The bounds also reduced the risk of producing extreme settings that could lead to unstable or unusable runs.

Table 3 Multiplier bounds used for hyperparameter exploration

Parameter group	**Lower bound**	**Default multiplier**	**Upper bound**
Geometry	0.2	1.0	5.0
Appearance	0.2	1.0	5.0
Densification	0.7	1.0	1.43

Geometry and appearance were given wider ranges because their selected learning rates could be explored through broader proportional changes. Densification was given a narrower range because its settings affect the creation, removal, and refinement of Gaussians. Fang et al. (2025) showed that threshold and density-related settings in 3DGS can be sensitive because changes in these settings may affect under-reconstruction, over-reconstruction, and final rendering quality. For this reason, the densification bounds were selected more conservatively than the geometry and appearance bounds.

These bounds were selected as practical limits for this study and were not treated as universal or optimal ranges for every 3DGS implementation or drone-image project. Geometry and appearance multipliers were first tested within a range of 0.5 to 2.0. Because some of the better relative quality scores occurred near the limits of this initial range, the range was expanded to 0.2 to 5.0 to explore a wider space. The values were sampled in log space so that smaller and larger proportional changes around the default multiplier of 1.0 were represented more evenly. The densification range was kept narrower because extending it to 0.5–2.0 produced substantially more Gaussians for many runs and noticeably increased training time.

4.2.2 Baseline Run

As shown in Step **2a of Figure 3**, one baseline run was completed for each model-development project. In this study, a **baseline run** refers to a 3DGS reconstruction completed using the default hyperparameter values listed in Table 2. It provided the reference result against which the exploration runs were compared.

A separate baseline run was completed for each project because reconstruction quality can differ between scenes, image sets, and acquisition conditions. So, each exploration run was compared only with the baseline from the same project, rather than with results from other projects. The quality comparison is explained in Section 4.2.4. The baseline run stored the reconstruction outputs, training logs, runtime, and final number of Gaussians. The same resized images and SfM outputs prepared in Section 4.1 were reused for both the baseline and exploration runs.

The same main processing settings were used in all baseline and exploration runs. This kept image size, training length, and evaluation timing consistent, which helped to keep the comparison focused on the tested multiplier combinations.

Table 4 Common settings used for baseline and exploratory 3DGS runs

Processing setting	**Value**
Maximum Size of image side	1,400 pixels
Maximum training iterations	5,000
Quality evaluation interval	1,000 iterations
Training-log interval	100 iterations
Densification active until	4,000 iterations
Gaussian hard cap	6,000,000

Quality evaluation was performed every 1,000 iterations because rendering the evaluation views and calculating the quality metrics added processing cost. Training logs were recorded every 100 iterations so that changes in loss, runtime behaviour, and Gaussian count could be followed more closely. Densification remained active until iteration 4,000. The remaining part of the run was then used to refine the existing Gaussian representation without continuing to create new Gaussians. A Gaussian hard cap of 6,000,000 Gaussians was set manually based on the available hardware. Beyond this limit, the GPU could not process some projects reliably, or a single run required too much memory and processing time. So, the Gaussian hard cap stopped runs with excessive Gaussian growth. A run stopped in this way is referred to as a **Gaussian hard cap** run. Only normally completed runs with usable quality results were included in the final training data.

A fixed checkpoint of 5,000 iterations was used for the baseline, exploratory, and later test runs. This value was not treated as the optimal convergence point for 3DGS. The original 3DGS study and the gsplat benchmark report results at shorter and longer

training budgets, including 7,000 and 30,000 iterations (Kerbl et al., 2023; Ye et al., 2025). The longer budget can improve final reconstruction quality, but it also increases the cost of every run.

The 3DGS runs were repeated many times for several projects. So the study had to consider the available GPU, memory, and processing time. The main purpose was comparative: different multiplier combinations were tested for the same project under the same training budget. The aim was to compare different multiplier combinations for the same project under the same training budget, rather than to achieve the highest possible quality from every run. For this reason, all runs were evaluated at 5,000 iterations. Using the same checkpoint kept the comparison fair because the results were not influenced by different training lengths.

The baseline and Ridge model-selected runs for all 12 test projects were further extended to 7,000 iterations to check whether the results remained similar. The same trained Ridge model, COLMAP sparse reconstructions, selected multiplier combinations, Gaussian hard cap of 6,000,000, and densification endpoint of 4,000 iterations were used. PSNR, SSIM, and LPIPS were evaluated at both the 5,000 and 7,000 iteration checkpoints to examine whether the relative comparison remained similar after further training. These results are presented in Section 6.4.4 and **Appendix 3E**. These findings support the use of 5,000 iterations as a practical fixed checkpoint for the main experiment. However, this checkpoint was not treated as full convergence, and identifying the best stopping iteration for each project was outside the scope of the thesis.

4.2.3 Exploration Runs

After the baseline run, exploration runs were carried out for each model-development project, as shown in **Step 2b of Figure 3**. An **exploration run** refers to a 3DGS reconstruction completed using a sampled multiplier combination applied to the default hyperparameter values. Fifteen exploration runs were planned for each project.

The purpose of these runs was not to fully optimize each project. Instead, they were used to create varied examples of how different multiplier combinations performed across different drone-image projects. These examples were later used to train the scoring models.

The number of exploration runs was selected as a practical compromise. More runs could provide greater coverage, but each additional configuration required a complete 3DGS run. The final choice is therefore balanced:

- variation across the multiplier ranges;
- the cost of each 3DGS run;
- the number of model-development projects;
- the available hardware and processing time;
- the need to create training data rather than find the best setting for every project.

Each exploration run used one multiplier combination containing three values:

$$m = [m_{geometry}, m_{appearance}, m_{densification}] \quad (2)$$

The three values were applied to the corresponding groups of default hyperparameters.

$$h_{tested} = m \times h_{default} \quad (3)$$

where $h_{default}$ represents the default hyperparameters, m represents one multiplier combination , and h_{tested} represents the resulting hyperparameter values used in the exploration run. Each multiplier in m was applied only to the default hyperparameters in its corresponding group. After the exploration run was completed, the multiplier combination used in that run was called a **tested multiplier combination**.

The allowed multiplier ranges were converted to log space before sampling:

$$z = log(m) \quad (4)$$

where m is a multiplier in its normal scale and z is its log-space value.

Log space was used to make sure the explored changes were proportional. On an ordinary scale, 0.5 and 2.0 do not appear equally distant from 1.0. However, they represent halving and doubling, which are balanced proportional changes. In log space:

$$log(0.5) = -log(2) \quad (5)$$

This allows increases and decreases around the default setting to be treated more evenly.

After sampling, each log-space value was converted back to the normal multiplier scale using the exponential function before it was applied to the default hyperparameters:

$$m = e^{z} \quad (6)$$

where e is the base of the natural logarithm and z is the sampled log-space value. The converted value m was then used as the geometry, appearance, or densification multiplier in the exploration run.

An exhaustive grid search was not practical for this study. Bergstra and Bengio (2012) showed that random hyperparameter search can be more efficient than grid search, particularly when the computational budget is limited. This supports the broader decision to use a limited search rather than evaluate every possible multiplier combination. However, the configurations in this study were generated using a Latin-hypercube-style procedure rather than ordinary random search.

The multiplier combinations were generated using bounded log-space **Latin Hypercube Sampling**. In this method, each variable range is divided into intervals, one value is sampled from each interval, and the sampled values are then combined across variables so that the available experiments cover the full range more evenly (McKay et al., 1979).

For each multiplier group, the log-space range was divided into 15 equal intervals. One value was randomly sampled from each interval. The geometry, appearance, and densification values were then shuffled separately and combined by their position. This produced 15 joint multiplier combinations for each model-development project as shown in Figure 4 below. The three groups were therefore explored together. Geometry, appearance, and densification were not tested through separate sets of runs.

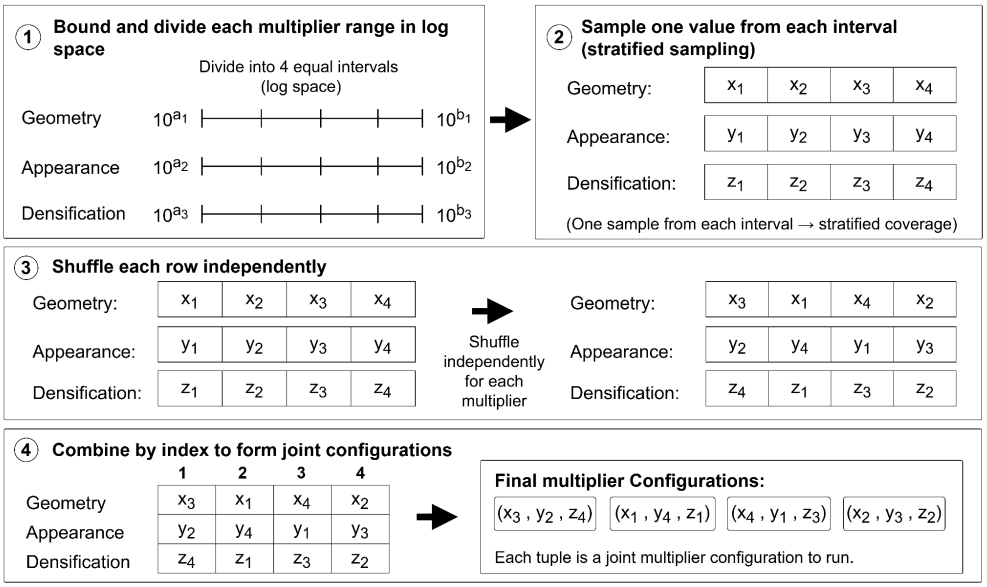


Figure 4 Example of bounded log-space Latin-hypercube-style sampling using four multiplier combinations.

Figure 4 shows four combinations to keep the example simple. However, the actual exploration used 15 multiplier combinations for each model-development project.

This sampling method helped to cover the full range of the three multiplier groups more evenly without requiring a complete grid search. A full grid using 15 values for each group would have produced 15×15×15=3,375 possible combinations. Instead, only 15 joint combinations were tested for each project, while values were still sampled across the full range of each multiplier group.

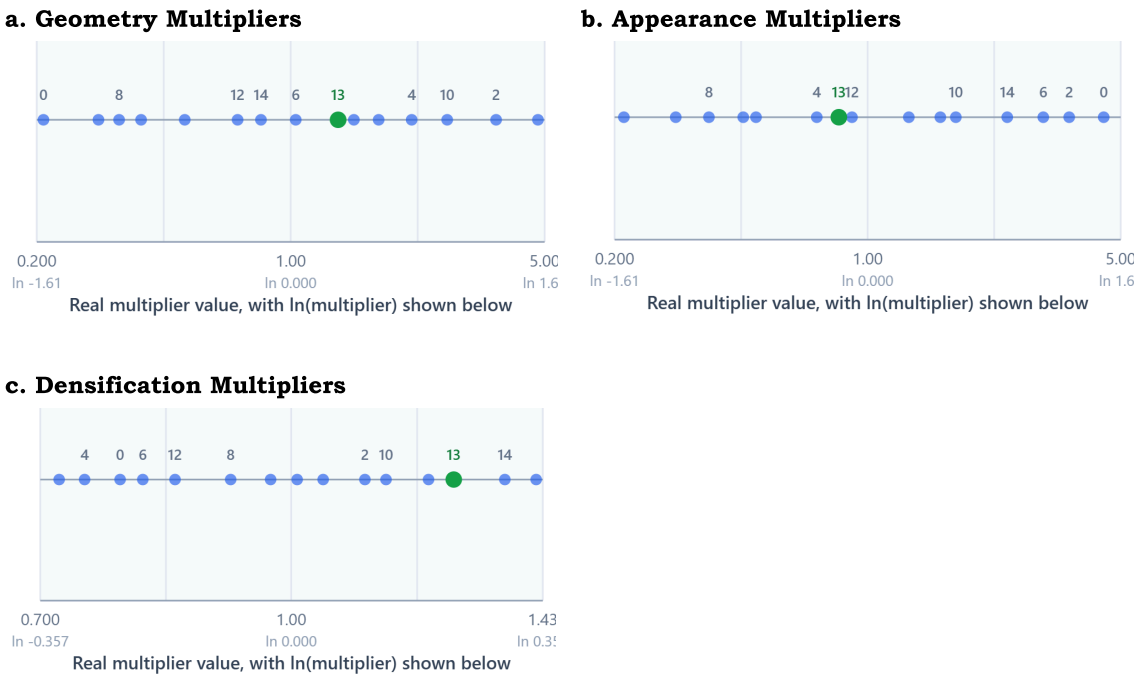


Figure 5 Distribution of the sampled geometry (a), appearance (b), and densification (c) multipliers within their defined log-space bounds.

The numbers above the points in Figure 5 show the shuffled order of the 15 sampled values for each multiplier group. One value was randomly sampled from each of the 15 equal intervals in log space. The green points represent the values used in the 13th joint multiplier combination, while ln denotes the natural logarithm. The figure shows that the sampled values were distributed across the full range of each multiplier group rather than being concentrated near the default multiplier of 1.0.

The baseline and exploration runs used the same resized images, SfM outputs, training length, and evaluation intervals. This helped to make the consistent comparison within each project.

The planned design included one baseline and 15 exploration runs for each of the 46 model-development projects: $46 \times (1 + 15) = 736$ runs. This was the planned number of reconstruction runs before final testing. It should not be confused with the selected final number of usable training rows, which is reported in Section 4.2.5. Completing the **736 runs** together with SfM took **9 days, 4 hours, and 9 minutes** using the hardware mentioned in section 3.3. This includes a four-minute cooldown period after each run to reduce overheating and allow the computer to cool before the next run begins.

4.2.4 Quality Evaluation

As shown in **Step 2c of Figure 3**, every completed exploration run was compared with the baseline from the same project. Reconstruction quality was measured using Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS). **PSNR** measures pixel-level difference between the rendered image and the reference image. A higher PSNR value usually means that the rendered image is closer to the reference image. **SSIM** measures how similar the image structures are by comparing brightness, contrast, and local patterns.

A higher SSIM value means that the rendered and reference images are structurally more similar. **LPIPS** measures perceptual difference using learned image features and is often used to compare visual similarity in a way that is closer to human perception. Unlike PSNR and SSIM, lower LPIPS values indicate better quality (Fang et al., 2025). Therefore, PSNR and SSIM were treated as positive indicators. LPIPS was first normalized as shown below in the same min–max way as the other metrics and then $1 - LPIPS_{norm}$ was used in the combined quality score.

The three metrics use different numerical ranges. They were therefore normalized before being combined. For a metric value v, **min–max normalization** was calculated as:

$$v_{norm} = \frac{v - v_{min}}{v_{max} - v_{min}} \tag{7}$$

The minimum and maximum were calculated separately for PSNR, SSIM, and LPIPS from the completed baseline and exploration runs used to prepare the training data. The normalized values were calculated as:

$$PSNR_{norm} = \frac{PSNR - PSNR_{min}}{PSNR_{max} - PSNR_{min}} \tag{8}$$

$$SSIM_{norm} = \frac{SSIM - SSIM_{min}}{SSIM_{max} - SSIM_{min}} \tag{9}$$

$$LPIPS_{norm} = \frac{LPIPS - LPIPS_{mean}}{LPIPS_{max} - LPIPS_{min}} \tag{10}$$

Because a lower LPIPS value indicates better quality, the normalized LPIPS value was reversed when the metrics were combined. The quality score for one run was calculated as:

$$Q_{quality} = \frac{PSNR_{norm} + SSIM_{norm} + (1 - LPIPS_{norm})}{3} \tag{11}$$

The three metrics were given equal weight. A higher $Q_{quality}$ value represented better combined reconstruction quality. Equal weights were used because no single quality metric was treated as more important across all projects, and application-specific weighting was outside the scope of the study.

The relative quality score was calculated by subtracting the baseline quality score from the quality score of the exploration run:

$$R_{quality} = Q_{quality}^{run} - Q_{quality}^{baseline} \tag{12}$$

A positive relative quality score indicated that the tested multiplier combination produced a higher combined quality score than the default baseline. A negative value indicated lower combined quality. A value close to zero indicated similar quality.

The relative quality score was created as an internal value for the scoring models. It combined PSNR, SSIM, and LPIPS into one learning target. This made it possible for

each scoring model to produce one score for a multiplier combination, which was more suitable for the limited amount of training data. The relative quality score was not used as the main measure in the final evaluation. Final test results were reported using the original PSNR, SSIM, and LPIPS differences compared with the baseline, as explained in Section 4.4.

4.2.5 Training Rows

The final part of Step 2 is shown as **Step 2d in Figure 3**. Each usable exploratory result was stored as one training row.

Each training row contained:

- the 10 project descriptors;
- one tested multiplier combination;
- the relative quality score for that exploration run.

The project descriptors remained the same for all exploration runs of one project. The tested multiplier combination and relative quality score changed from one run to another.

A total of 736 runs were completed for the 46 model-development projects. Of these, 699 completed normally and 37 reached the Gaussian hard cap. The 699 normal runs included 46 baseline runs and 653 exploration runs. Since the baseline runs were used only to calculate the relative quality score, the final training data contained **653 usable training rows**.

A second training-data version was also tested by keeping the 37 hard-cap runs and assigning each of them a penalty score. The penalty was set to the lowest relative quality score found among the normally completed exploration runs. This represented a poor outcome for runs that exceeded the Gaussian hard-cap limit.

The final Ridge Regression and MLP scoring models used only the 653 normally completed exploratory rows because this version produced better results. The comparison with the penalty-based training data for hard-cap runs is presented in Section 6.4.3.

The 3DGS runs to prepare training rows and scoring-model training were kept as separate stages. All reconstruction results were first completed and stored as training rows. Then these training rows were used to train the scoring models. This avoided interrupting long reconstruction runs and kept the 3DGS processing stage separate from model training. The detailed model training process is described in the next section.

4.3 Scoring-Model Training

This section explains **Step 3 of Figure 3**. The 653 training rows prepared in Section 4.2 were used to train two scoring models: Ridge Regression and a multilayer perceptron (MLP).

The final training data contained 653 rows, but these rows came from only 46 model-development projects. The project descriptors were repeated across the exploration runs of the same project, while only the tested multiplier combination and relative quality score changed. Therefore, the effective variation in project characteristics was limited to 46 independent projects. More flexible neural-network models were not selected as the main approach because they generally require more independent and diverse examples to learn stable relationships. However, a small MLP was retained as a nonlinear comparison model to examine whether a nonlinear model could provide better predictions from the same inputs.

Ridge Regression was selected as the main scoring model because it provides a simple regularized approach for limited data and potentially related inputs. Chaibub Neto et al. (2014) compared Ridge Regression with LASSO and Elastic Net in limited-sample settings. Ridge substantially outperformed LASSO and was competitive with Elastic Net, especially when the sample size was small and useful information was spread across many input variables. This was relevant to the present study because the number of independent projects was limited, while several project descriptors and descriptor–multiplier interaction terms could also be related.

The shared model inputs and learning target are defined in Section 4.3.1, followed by the individual Ridge Regression and MLP model designs.

4.3.1 Shared Model Input and Learning Target

Each training row contained:

- 10 standardized project descriptors;
- one tested multiplier combination containing geometry, appearance, and densification multipliers;
- one relative quality score.

As described in Section 4.2.3, the multiplier values were converted to log space before being used by the scoring models. The resulting log-transformed multiplier combination is represented by z.

The models were not trained to predict multiplier values directly because no confirmed optimal multiplier combination was available for each project. Instead, the models learned to estimate the quality of a complete multiplier combination:

$$\hat{R}_{quality} = f(x, z) \tag{13}$$

where x represents the standardized project descriptors, z represents the log-transformed multiplier combination, and $\hat{R}_{quality}$ is the predicted quality score.

The observed **relative quality score** was calculated from a completed exploration run, whereas the **predicted quality score** was the value estimated by a scoring model. During final testing, each model scored several candidate multiplier combinations. The combination with the highest predicted quality score was selected, as explained in Section 4.4.

4.3.2 Ridge Regression Scoring Model

Ridge Regression was used as the main scoring model. It extends least-squares regression by adding a regularization term, which shrinks coefficient values and helps keep their estimates stable when input variables are related (Hoerl & Kennard, 1970).

Using the shared formulation defined in Section 4.3.1, the Ridge model estimated the predicted relative quality score for a project and a tested log-space multiplier combination. A model using only the project descriptors and multiplier values would mainly represent increasing or decreasing trends. So, the highest predicted score would often occur near the multiplier bounds. However, the best result could also occur between the lower and upper multiplier bounds. The figure below illustrates simple relationships that Ridge Regression can represent and a more complex relationship that may require a nonlinear model.

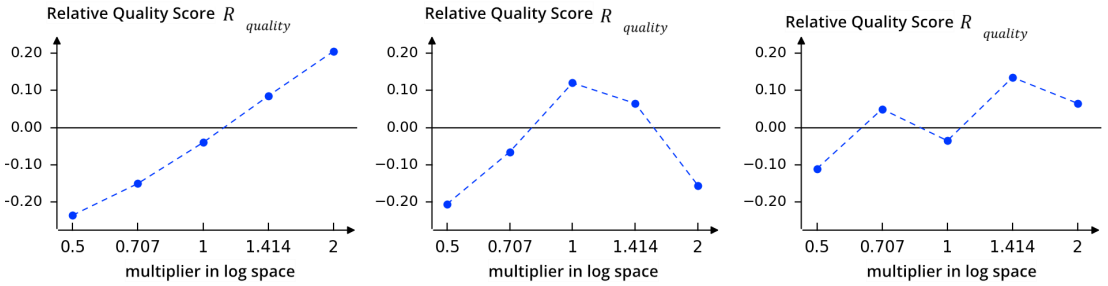


Figure 6 Examples of possible relationships between a multiplier and the relative quality score.

The first relationship can be represented using the multiplier value. The second can be represented using both the multiplier and squared multiplier values, which allows the model to represent a simple U-shaped or inverted U-shaped curve. The M-shaped relationship in the third example cannot be represented using only these terms because it changes direction more than once. A more flexible nonlinear model, such as the MLP described in Section 4.3.3, is needed for this type of pattern. These examples show why the Ridge input was expanded beyond the original descriptors and log-transformed multiplier values. For one training row i, the Ridge input vector was defined as:

$$\phi(x_i, z_i) = [1, x_i, z_i, z_i^2, x_i z_i]^T \tag{14}$$

where x_i represents the standardized project descriptors for training row i, and z_i represents the log-transformed multiplier combination tested in that row.

The vector contained:

- one intercept;
- 10 standardized project descriptors;
- three multiplier values;
- three squared multiplier values;
- 30 descriptor–multiplier interactions.

This produced 47 input values. The squared values z_i^2 allowed the model to represent simple U-shaped or inverted U-shaped patterns. The interaction values $x_i z_i$ allowed the effect of the tested multiplier to depend on the project descriptors. For example, the effect of an appearance multiplier could differ between projects with high and low texture density. Here, $x_i z_i$ represents all pairwise products between the 10 project descriptors and the three log-transformed multiplier values, producing 30 interaction terms.

The predicted quality score was calculated as:

$$\hat{R}_{quality,\,i} = \phi(x_i,\, z_i)^T \hat{\theta} \tag{15}$$

where $\phi(x_i,\, z_i)$ is the 47-value Ridge input vector for training row i, and $\hat{\theta}$ contains the learned Ridge coefficients.

Hoerl and Kennard (1970) expressed Ridge Regression by adding a regularization term to the least-squares system. Following the same principle, the coefficients in this study were estimated as:

$$\hat{\theta}_{\lambda} = \left(\sum_{i=1}^{N} \phi_i \phi_i^T + \lambda I\right) \left(\sum_{i=1}^{N} \phi_i R_{quality,\,i}\right) \tag{16}$$

Here, N is the number of training rows, $\phi_i = \phi(x_i,\, z_i)$, $R_{quality,\,i}$ is the observed relative quality score in training row, λ controls the regularization strength, and the matrix P applies regularization to all learned coefficients except the intercept. The matrix P was defined as:

$$\mathrm{P} = \mathrm{diag}(0, 1, 1, \ldots, 1) \tag{17}$$

Here P is a diagonal matrix whose first diagonal value is 0 and all remaining diagonal values are 1. Therefore, P is a modified identity matrix that excludes the intercept from regularization. The intercept gives the model its reference prediction when all standardized descriptors and log-space multiplier values are zero. In this condition, the descriptors are at their mean values and the multipliers are equal to 1.0. Regularizing the intercept would unnecessarily pull this reference prediction towards zero, so only the remaining coefficients are regularized.

The following regularization values were tested:

$$\lambda \in \{0.1, 0.5, 1.0, 2.0, 5.0, 10.0, 20.0\} \quad (18)$$

For each λ, the mean squared error was calculated from the 653 training rows:

$$MSE(\lambda) = \frac{1}{N}\sum_{i=1}^{N}(\phi_i^T \hat{\theta}_\lambda - R_{quality,i})^2 \quad (19)$$

where N is the number of training rows, ϕ is the Ridge input vector, $\hat{\theta}_\lambda$ is the coefficient vector fitted for the tested value of λ, and $R_{quality,i}$ is the observed relative quality score. The value with the lowest mean squared error was selected. In the final experiment, the selected value was $\lambda = 0.1$. The final Ridge Regression model was then fitted using all 653 training rows.

The learned coefficients were later examined as indicators of the patterns found by the model. They were not treated as proof that a project descriptor directly caused a change in reconstruction quality.

4.3.3 MLP Scoring Model

The MLP was used to test whether a small nonlinear model could learn patterns that were not captured by Ridge Regression. Its size was kept small because the number of training rows and independent projects was limited.

The MLP used the same 47 input values as Ridge Regression. These included the standardized project descriptors, multiplier values, squared multiplier values, and descriptor–multiplier interactions. Ridge Regression used expanded input terms because it requires explicit interaction and squared terms to model nonlinear effects. The MLP was tested with the same expanded representation to keep the scoring formulation comparable with Ridge. Although the network was small, the training rows came from a limited number of independent projects. A low dropout rate of 0.2 was therefore used as an additional safeguard against overfitting by randomly deactivating some hidden-layer units during training (Srivastava et al., 2014).

Its architecture was: 47 → 16 → 1. The network contained one hidden layer with 16 units and one output for the predicted quality score.

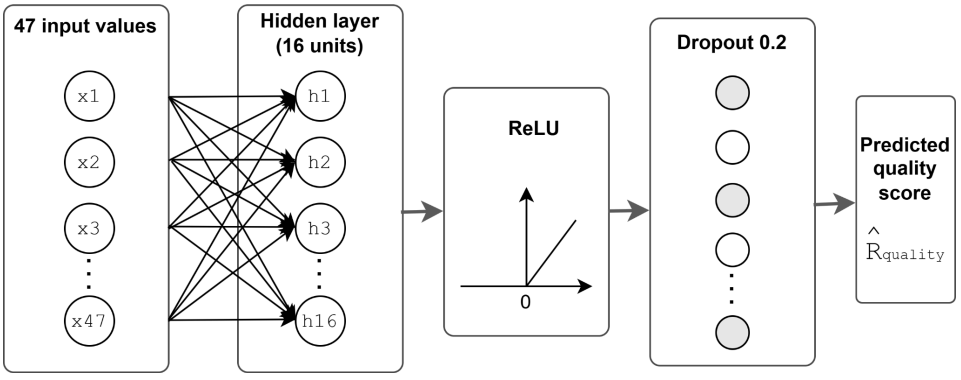


Figure 7 Architecture of the shared MLP scoring model.

Table 5 MLP scoring-model training settings

Sn	Setting	Value	Sn		
1	Architecture	47 → 16 → 1	6	Weight decay	0.001
2	Activation	ReLU	7	Maximum epochs	1,000
3	Dropout	0.2	8	Early-stopping patience	50 epochs
4	Optimizer	Adam	9	Training and validation split	80% / 20% by project
5	Learning rate	0.001	10	Random seed	42

The 46 model-development projects were divided into training and validation groups using an 80%/20% project-level split. All exploration rows from one project were kept in the same group. This was important because rows from the same project shared the same descriptors and scenes. A random row-level split could place rows from the same project in both groups and give an overly positive validation result. The validation projects were used only during MLP training and remained separate from the 12 final test projects.

Training stopped when the project-level validation loss did not improve for 50 consecutive epochs. The model weights from the epoch with the lowest validation loss were retained. A fixed **random seed** of 42 was used because MLP training includes random steps, including the project-level train-validation split, the initial network weights, and dropout during training. This made the training process reproducible when the same data and settings were used.

4.4 Model Testing and Evaluation

This section explains **Step 4 of Figure 3**. The trained Ridge Regression and MLP scoring models were applied to the 12 final test projects. These projects were not used during training-data preparation or model training.

A **candidate multiplier** combination refers to one possible set containing a geometry, appearance, and densification multiplier. Each trained scoring model evaluated the candidate combinations and selected the one with the highest predicted relative quality score. This selected set is referred to as the **model-selected multiplier** combination. A **3DGS test run** refers to an actual reconstruction completed for a final test project using either the default settings or a model-selected multiplier combination. A run completed using a selected combination is referred to as a **model-selected run**.

The project descriptors for each test project were prepared and standardized using the procedure described in Section 4.1. The multiplier bounds defined in Table 3 were reused to prepare the candidate multiplier combinations. The candidate-generation procedure followed the same general structure as the Latin-hypercube-style sampling described in Section 4.2.3. However, instead of dividing each log-space range into intervals and randomly sampling one value from each interval, the test procedure used

30 regularly spaced values within each multiplier range. Then, 30 values for the geometry, appearance, and densification groups were then independently shuffled. Values at the same shuffled position were combined, producing 30 complete candidate multiplier combinations. Each combination contained one geometry, one appearance, and one densification multiplier and followed the same input structure used during model training. Thirty candidates were used to provide denser coverage than the exploration stage while keeping the candidate set compact and consistent across all test projects.

Each candidate combination was scored separately by the Ridge Regression and MLP models:

$$\hat{R}_{quality}(x, z) = f(x, z) \tag{20}$$

Here, f represents the trained Ridge Regression or MLP scoring model, x represents the standardized project descriptors, z represents one log-transformed candidate multiplier combination, and $\hat{R}_{quality}$ is the predicted quality score produced by the scoring model.

The combination with the highest predicted quality score was selected for each model:

$$z^{*} = \arg\max_{z \in Z} \hat{R}_{quality}(x, z) \tag{21}$$

Here, Z is the set of candidate multiplier combinations, and z^{*} is the combination with the highest predicted quality score. The selected values were then converted from log space back to their normal multiplier scale before being applied to the default hyperparameters.

The Ridge Regression and MLP models could select different combinations for the same test project because they learned the scoring relationship differently.

Three 3DGS test runs were completed for each final test project:

- one baseline run using the default hyperparameters
- one Ridge model-selected run
- one MLP model-selected run

All runs used the same images, SfM outputs, 5,000-iteration checkpoint, and processing settings described in Sections 4.1 and 4.2. This helped ensure that the evaluation conditions remained consistent, so the main planned difference between the baseline and model-selected runs was the applied hyperparameter combination. The selected multiplier values, predicted quality scores, PSNR, SSIM, LPIPS, runtime, and final number of Gaussians were recorded for each test run.

Each model-selected run was compared with the baseline from the same test project. Final performance was reported using the PSNR, SSIM, and LPIPS values rather than the internal relative quality score.

The changes in the quality were calculated as:

$$\Delta PSNR = PSNR_{model} - PSN_{base} \quad (22)$$

$$\Delta SSIM = SSIM_{model} - SSIM_{base} \quad (23)$$

$$\Delta LPIPS = LPIPS_{base} - LPIPS_{model} \quad (24)$$

n these equations, the model value refers to either the Ridge-selected run or the MLP-selected run. A positive $\Delta PSNR$, or $\Delta SSIM$ means that the model-selected run performed better than the baseline. For LPIPS, the subtraction order was reversed because lower values indicate better quality. Therefore, a positive ΔLPIPS also means an improvement.

The evaluation mainly examined whether the Ridge and MLP model-selected runs improved or maintained reconstruction quality compared with the default baseline. Runtime, Gaussian count, convergence behaviour, and visual comparisons were reported as supporting results.

5 Platform Development

To support the workflow described in Chapter 4, **Bimba3D-re** was developed as part of this thesis as an open-source local platform and made publicly available through GitHub. The platform supports both practical drone-based 3DGS processing and the research workflow used in this thesis. Users can prepare their image projects, create 3DGS models, apply trained scoring models, and review the results.

The platform has two connected levels. The **project** level contains the pages and tools used to process one drone-image dataset. It includes image preparation, SfM processing, and 3DGS training. The **Research Workflow** is the higher-level part of the platform which helps to run multiple projects as a batch process. It applies shared settings to several projects and collects their results for training-data preparation, scoring-model training, testing, and evaluation.

Bimba3D-re can be run as a local web application. It uses a Python FastAPI backend and a React and TypeScript frontend. COLMAP is used for SfM processing, while gsplat is used for 3DGS training. Reconstruction outputs and trained models are stored in their corresponding project folders. Project information, workflow records, run configurations, metrics, and logs are stored in JSON files. Most experiments in this thesis were run directly on the local computer, while Docker is also supported as an optional deployment method for users who prefer a containerized setup. The software and hardware requirements are described in Chapter 3. The platform interface can run on a standard computer, but a CUDA-enabled NVIDIA GPU is recommended for COLMAP and 3DGS processing. Small projects may run with 4–6 GB of VRAM and 16 GB of system RAM, while at least 8 GB of VRAM and 32 GB of RAM are recommended for larger projects and repeated experiments. The actual requirements depend on the image size, project size, number of Gaussians, and selected training settings.

5.1 Platform Overview and Project-Based Processing

The dashboard provides access to the Projects, Research Workflow, and Comparison pages as shown in the figure below.

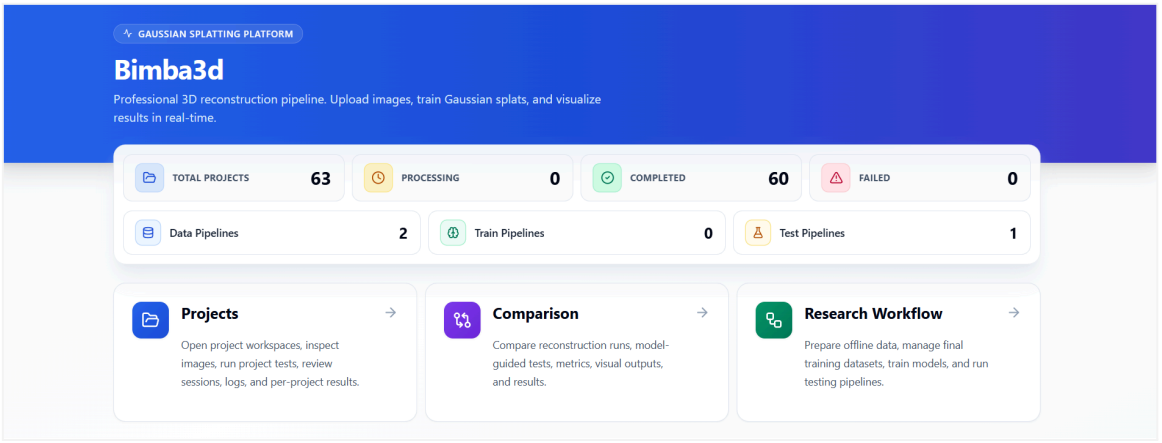


Figure 8 Bimba3D-re homepage.

The Projects page is used for processing individual drone-image datasets. A user can create a project, upload images, configure a run, and start the required processing stages. Within each project, the platform can prepare the images, run COLMAP/SfM, and train a 3DGS model using *gsplat*. It stores the processing status, logs, run configuration, quality metrics, Gaussian count, previews, and reconstruction outputs. Several runs can be stored within the same project and compared using their metrics and visual outputs on the comparison page.

The project-based flow can be used independently of the Research Workflow. A user can therefore create a practical 3DGS reconstruction using default or manually selected settings. In this case, the platform first prepares a baseline run and then completes a model-selected run using the multiplier combination selected by the model.

A project configuration includes an option for training data enrichment. When the "update training data" option is enabled, the platform stores the project descriptors, multiplier values, and quality results from the fixed 5,000-iteration checkpoint. Training rows can now be rebuilt using the Rebuild button in Research Workflow's Training data page. After the training rows are rebuilt, the new model can be trained and tested with updated training rows. This allows future drone-image projects to add new examples to the available training data. Also, for a better update of training data, the platform can be hosted on a server in the future. Public users can then start using this pipeline to generate results. They can set the project configuration to update training data. This can help collect more training datasets from real 3DGS runs from the platform. Currently, the platform is only a local installation from GitHub because it needs a high-end server to support drone-image processing.

The platform also provides live monitoring during project processing. The interface shows the current stage, progress, processing messages, logs, and run status while SfM and 3DGS tasks are running. Processing can be stopped, resumed, or restarted as needed.

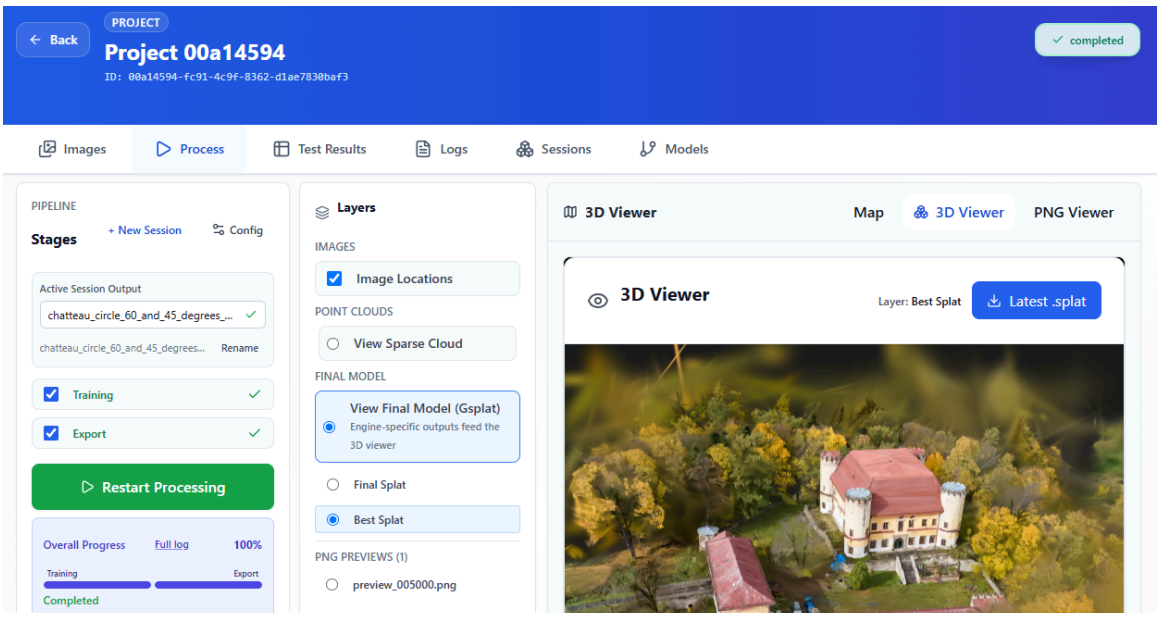


Figure 9 Project-based interface for image preparation, SfM processing, 3DGS training, and run-level result inspection.

5.2 Research Workflow

The Research Workflow is the higher-level part of Bimba3D-re. It groups several project-level runs under one shared pipeline and follows the four main steps of the method described in Chapter 4: Project Descriptor Extraction and SfM Processing, Prepare Training Data, Train Model, and Test and Evaluate Model.

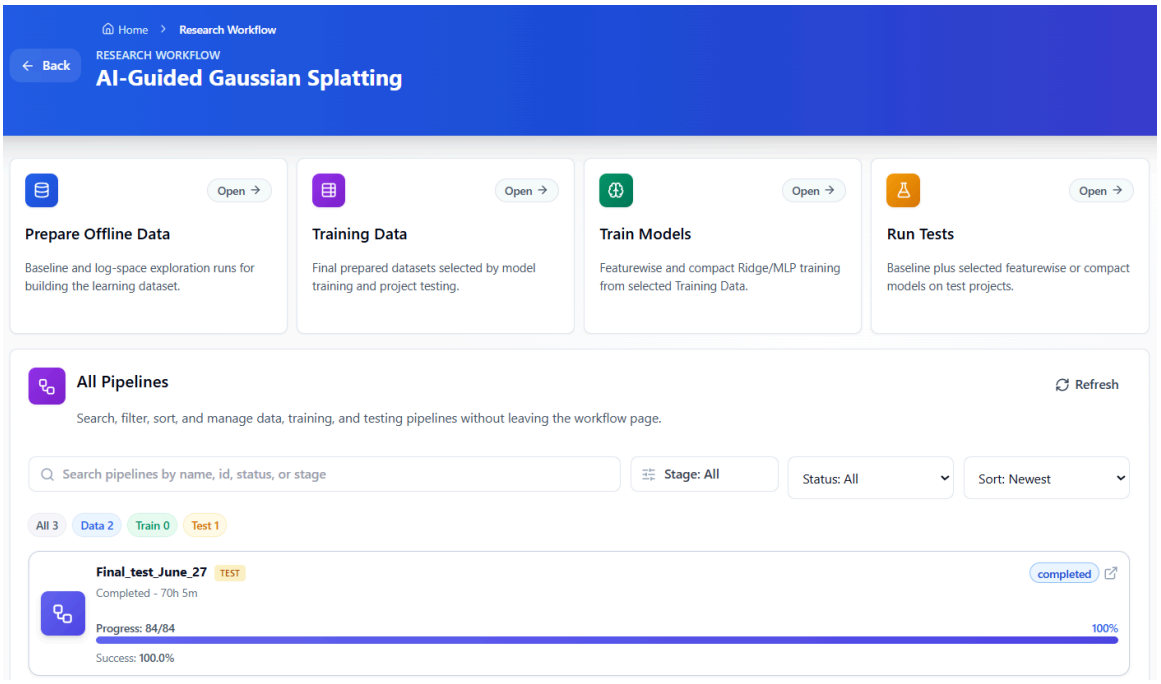


Figure 10 Research Workflow page showing the four main stages implemented in Bimba3D-re.

The workflow page provides access to these four stages and also lists the related pipelines and their current status. The following subsections explain how each stage was implemented in the platform.

5.2.1 Project Descriptor Extraction and SfM Processing

This part of the platform implements Step 1 of the method described in Section 4.1. The required processing is carried out inside each project workspace. The original images and available metadata are used to prepare the project descriptors. The platform also performs image resizing and COLMAP/SfM processing. The resulting camera information and sparse reconstruction are stored with the project and reused during later 3DGS runs. The descriptor calculation and standardization procedures were described in Section 4.1. The prepared descriptors and SfM outputs are used later during training-data preparation and model testing.

5.2.2 Training Data Praparation

The **Prepare Offline Data** and **Training Data** pages implement **Step 2** of the workflow. The Prepare Offline Data page allows the user to select several model-development projects and apply one shared pipeline configuration. The configuration includes the baseline run, number of exploration runs, multiplier bounds, log-space sampling

settings, runtime configurations, and cooldown period. The platform then starts the required project-level runs and manages them as one **pipeline**.

Live monitoring is available throughout this stage. The interface shows the active project, current run, processing stage, overall progress, completed runs, hard-cap runs, elapsed time, logs, and worker messages. The pipeline can be started, paused, resumed, stopped, restarted, or used to retry failed runs. This was important because offline data preparation involved many long 3DGS runs. A failed project run could be repeated without restarting the whole pipeline.

The baseline results are collected first. The exploration runs then apply the sampled geometry, appearance, and densification multiplier combinations described in Section 4.2. The saved multiplier values and run-level results can be inspected through the pipeline detail page.

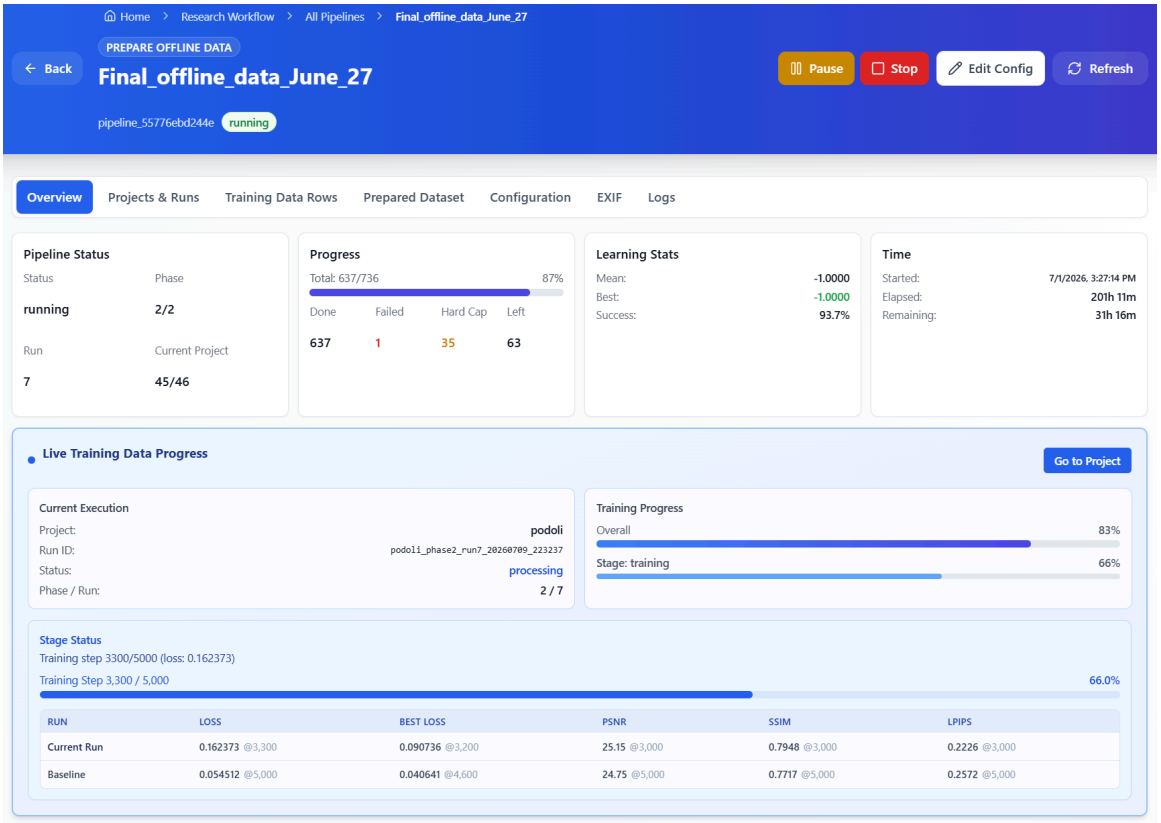


Figure 11 Offline data preparation page used to configure, monitor, and collect repeated project-level runs.

After the offline runs are completed, the **Training Data** page builds the final dataset from the stored dataset to create the Training rows, as shown in Figure 12 below. The table allows the user to inspect the project, run, multiplier values, descriptors, relative quality score, and hard-cap status. This helps verify the rows before model training.

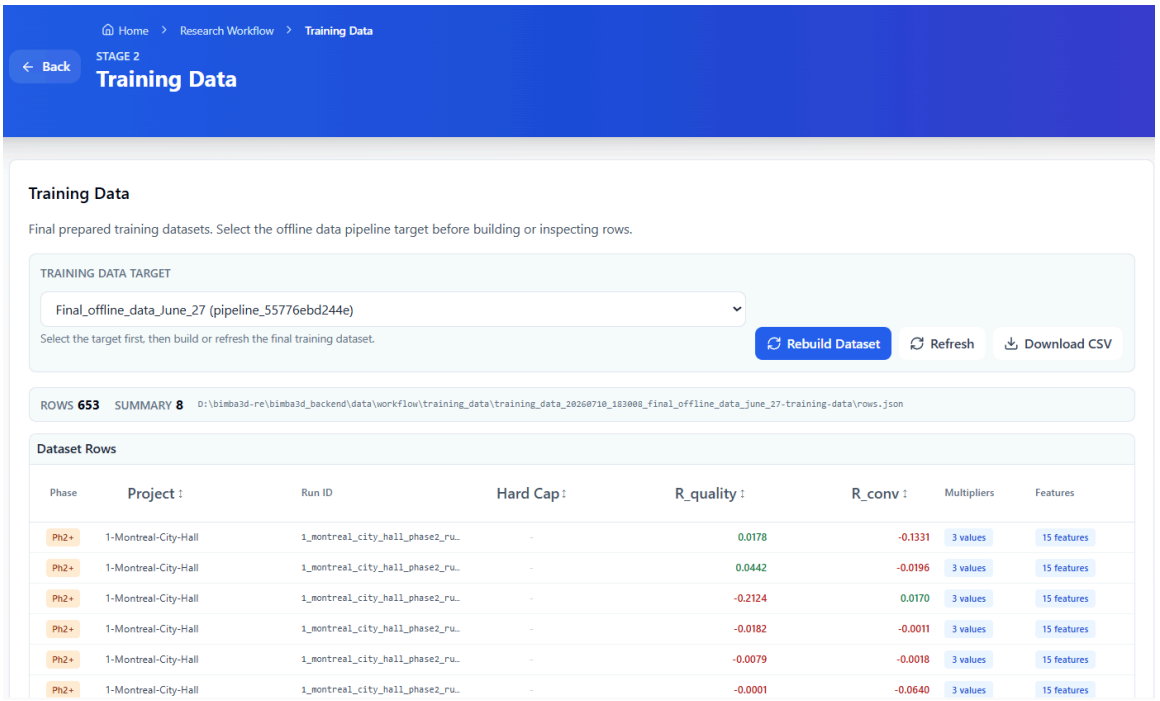


Figure 12 Training Data page for preparing training data rows using the offline data generated through the Prepare Offline Data page.

The dataset can be rebuilt when new results are available and can also be downloaded as a CSV file. The platform includes an optional setting for adding penalty rows from hard-cap runs. Only rows from normally completed runs were used to train the final models. Rows from hard-capped runs were excluded, as explained in Section 4.2.5.

5.2.3 Scoring Model Training

The **Train Models** page is the implementation of **Step 3** as shown in Figure 3. A prepared Training Data is selected before model training begins. The user can then train a Ridge Regression or MLP scoring model using the settings described in Section 4.3. Each trained model is stored together with its model type, source Training Data, configuration, and available training results. Registered models can later be inspected and reused in testing pipelines. Standalone notebooks are also provided for Ridge Regression and MLP training. After Training Data has been prepared, the notebooks can be used to load the dataset, train the model, display model statistics and intermediate outputs, and save the trained model file. The notebooks support model training and candidate multiplier scoring, but they do not complete the final 3DGS reconstruction using the model-selected multiplier combination. To complete the 3DGS-based test, the trained model must be uploaded to the platform and applied through either a workflow test pipeline or a single-project test, which is described in Section 5.2.4 below.

5.2.4 Model Testing and Evaluation

The platform provides two ways to test a trained scoring model. The first is the **Run Tests** stage in the Research Workflow. This method is used as a batch process to test one or more trained models across several unseen projects. The user selects the test projects and the registered Ridge Regression or MLP models from pipeline configuration.

The pipeline then prepares the baseline runs, scores the candidate multiplier combinations, and starts the model-selected 3DGS runs one by one.

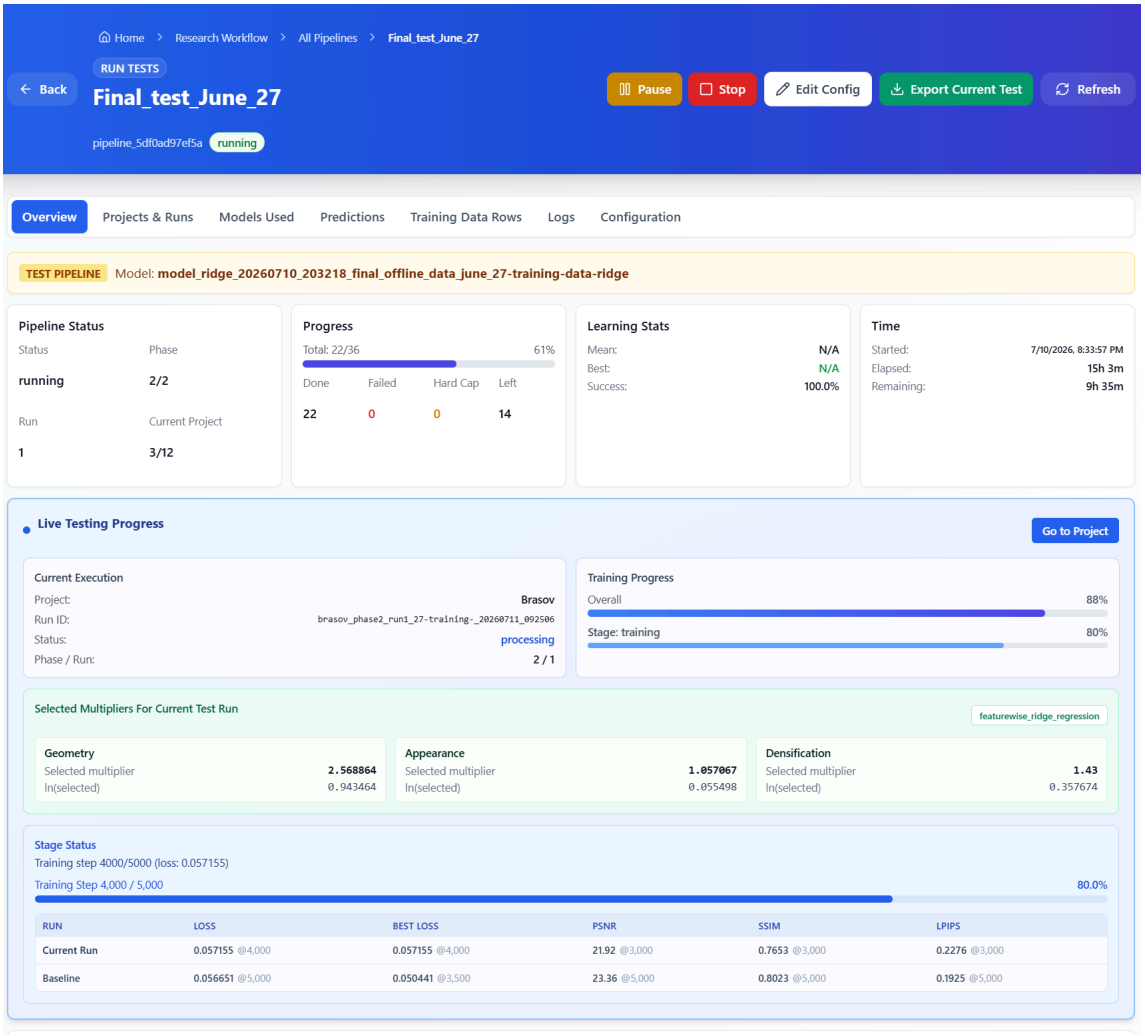


Figure 13 Run Tests Page with live testing progress and live comparison of quality metrics at different steps the 3DGS runs.

The second method is available inside a single project. A user can create a baseline run, select a registered scoring model, and complete a model-selected run using the combination selected by that model. The project configuration can also enable data enrichment so that the completed result can add the row in the selected training rows. The updated training data can be used to rebuild a new model and test them.

In both methods, the model first calculates predicted relative quality scores for the candidate multiplier combinations. The model-selected multipliers for geometry, appearance, and densification are then applied to the default hyperparameters and passed to the project-processing layer, where the model-selected 3DGS run is completed. The platform stores the predicted relative quality score along with the final PSNR, SSIM, and LPIPS values measured after the 3DGS run.

Live monitoring is also available during testing. The platform shows the current project, model, run status, progress, logs, and worker messages. Testing pipelines can be paused, resumed, stopped, restarted, or used to retry failed runs.

When the test pipeline is running, the page displays charts showing how the models scored the candidate multiplier combinations and which multiplier values were selected. It also compares each model-selected run with the corresponding baseline. This includes changes in PSNR, SSIM, and LPIPS, overall model performance, and project-level results. Runtime, Gaussian count, processing logs, and individual test results can also be reviewed on this page. The charts can be exported in PNG or SVG

The **Comparison** page allows the users to review two selected runs side-by-side. Their loss, PSNR, SSIM, and LPIPS curves can be displayed in the same charts. This makes differences in training behaviour and final quality easier to examine. The page also includes a swipe comparison of the image previews produced at the 5,000-iteration checkpoint. A slider can be moved between the two images to compare them visually. These tools support both metric-based and visual analysis of the reconstruction results.

5.3 Availability and Reuse

The source code for Bimba3D-re is publicly available on GitHub under the **GNU General Public License version 3**. The platform can run locally on Windows and Linux, with Docker provided as an optional setup. The research output files are maintained in a separate GitHub repository because of their large file size.

The platform and notebooks provide a reusable implementation of the research workflow. Other users can process their own drone images, create 3DGS models, train or import scoring models, score candidate multiplier combinations, complete model-selected runs, enrich the training rows, and rebuild the scoring models. This supports both reproduction of the thesis experiments and future improvement of the scoring models.

6 Results

This chapter presents the results of training-data preparation, scoring-model training, and evaluation on the unseen test projects. The relative quality score is reported only for the training data, while final performance is evaluated using PSNR, SSIM, and LPIPS differences between the model-selected and baseline runs. Runtime, Gaussian count, and visual comparisons are included as supporting results. The complete project-level outputs and analysis files are available in the separate research-outputs GitHub repository.

6.1 Summary of Results

The main results of the final Ridge Regression and MLP evaluations are summarized in Table 6. The quality means and medians were calculated only from normally completed runs because the Gaussian hard-cap runs were stopped before final PSNR, SSIM, and LPIPS values could be produced. **Normally completed runs** are those that successfully executed the 3DGS process without reaching the 6,000,000 **Gaussian hard-cap** safety limit and produced final quality metrics. Table 6 provides the summary of the results for both Ridge Regression and MLP and check marks indicate better value for each comparison.

Table 6 Summary of the main experimental results

Result	Ridge Regression	MLP
Model-development projects	46	46
Usable training rows	653	653
Test projects	12	12
Normally completed runs	11/12 ✓	10/12
Hard-cap runs	1 ✓	2
Mean / Median PSNR difference (only completed runs)	+0.2536 dB / +0.1413 dB	+0.2614 dB / +0.2283 dB ✓
Mean/Median SSIM difference (only completed runs)	+0.0428 / +0.0311	+0.0503 / +0.0322 ✓
Mean / Median LPIPS improvement (only completed runs)	+0.0835 / +0.0711	+0.1015 / +0.0750 ✓
PSNR improved	8/11 ✓	6/10
SSIM improved	11/11 ✓	9/10
LPIPS improved	11/11 ✓	9/10
All three metrics improved	8/11 ✓	6/10
Mean additional Gaussians	+953,204 ✓	+1,410,348
Mean additional runtime	+167 s ✓	+222 s

The Ridge Regression model completed 11 of the 12 test projects normally, while one model-selected run reached the Gaussian hard cap and did not produce final quality

metrics. Among the completed runs, PSNR improved in eight projects, while SSIM and LPIPS improved in all 11 projects. All three metrics improved together in eight projects.

The MLP model completed 10 of the 12 test projects normally, while two model-selected runs reached the Gaussian hard cap. Among the completed runs, PSNR improved in six projects, while SSIM and LPIPS improved in nine projects. All three metrics improved together in six projects.

Figure 14 presents the number of test projects that improved, did not improve, or reached the Gaussian hard cap for each quality metric. Hard-cap runs are included in the total number of test projects but are not counted as improved or non-improved because no final quality metrics were produced.



Figure 14 Summary of improved, not-improved, and Gaussian-hard-cap test runs for (a) Ridge Regression and (b) the MLP.

The **model-selected** runs generally required more processing resources than the corresponding baseline runs. The normally completed Ridge runs produced approximately 953,204 additional Gaussians and required 167 seconds more runtime on average. The normally completed MLP runs produced approximately 1,410,348 additional Gaussians and required 222 seconds more runtime on average. These values are reported as supporting results because the models were trained to estimate reconstruction quality rather than directly optimize runtime or Gaussian count.

Overall, the results show that the proposed workflow achieved its main objective of using information from previous projects to guide hyperparameter selection for unseen projects. Both scoring models produced positive mean changes in PSNR, SSIM, and LPIPS among their normally completed runs. However, the recommendations did not improve every project, and some model-selected runs produced higher Gaussian counts or reached the Gaussian hard-cap. The following sections present the detailed training-data results, model-training results, project-level quality differences, and supporting comparisons.

6.2 Offline Data Preparation and Training Data

Offline data was prepared using 46 model-development projects as described in Section 4.1. Each project was processed using one baseline run and 15 exploration runs. This produced 736 runs in total, consisting of 46 baseline runs and 690 exploration runs.

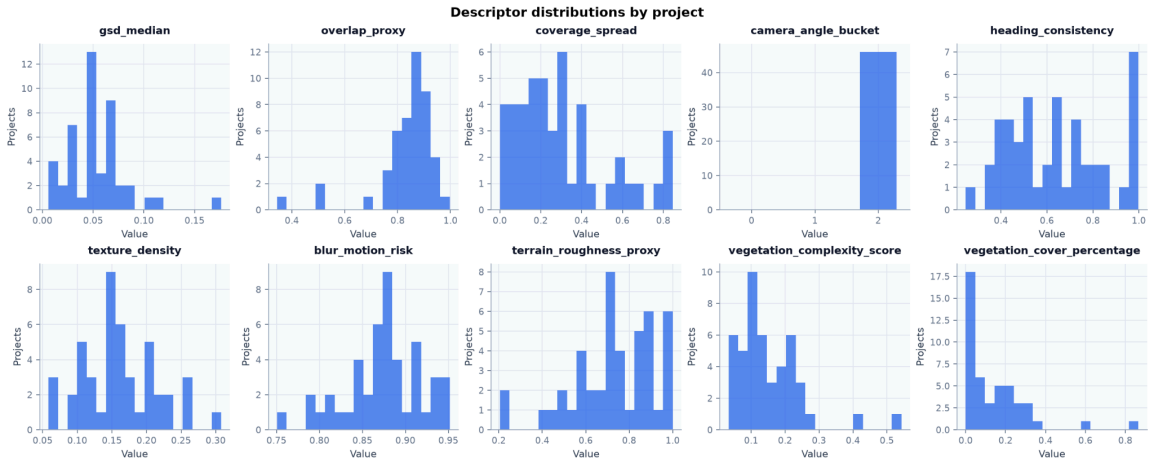


Figure 15 Distribution of the 10 project descriptors. Where the x-axis shows descriptor values, and the y-axis represents the number of model-development projects.

Estimated overlap was mainly concentrated toward the higher end of its range, while vegetation cover was concentrated toward the lower end. Terrain roughness included many higher values, and the camera angle showed no variation because all model-development projects were assigned to oblique/mixed category. The distributions also show that some descriptor ranges were represented by only a small number of projects, which may have limited model learning for less common project conditions.

All 46 baseline runs were completed successfully. Among the exploration runs, 653 completed normally and 37 reached the Gaussian hard-cap. These hard-cap runs were intentionally stopped by the predefined safety limit and were not classified as failed runs. Completing the 736 runs together with SfM took 9 days, 4 hours, and 9 minutes, including the 4-minute cooldown period applied between consecutive runs.

Table 7 Offline data preparation and training-data

Result	**Value**
Model-development projects processed	46
Baseline runs completed normally	46
Exploration runs completed normally	653
Hard-cap reached exploration runs	37
Total runs	736
Failed runs	0
Final usable training rows	653
Total processing time including SfM, baseline runs, and exploration runs	9 days 4 hours 9 minutes

The baseline runs were used as project-specific references for evaluating the exploration runs. They were not added as separate rows to the final training data. Each normally completed exploration run produced one training row containing the project descriptors, the tested multiplier combination, and the relative quality score. Further details on the training-row structure and score calculation are provided in Sections 4.2.4 and 4.2.5.

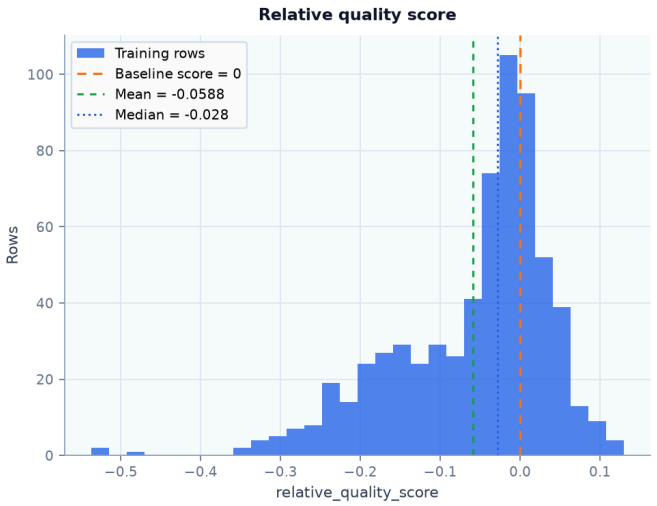


Figure 16 Distribution of relative quality scores for the 653 normally completed exploration runs. The dashed orange line marks a relative quality score of zero, while the green and blue lines show the mean and median values.

The relative quality scores ranged from −0.5373 to +0.1297, with a mean of −0.0588 and a median of −0.0280. Of the 653 training rows, 191 had positive scores and 462 had negative scores. A positive score indicates that the tested multiplier combination performed better than the project-specific baseline according to the internal score, while a negative score indicates a weaker result. Most scores were close to zero, although the distribution contained fewer large positive and negative values.

The training rows covered multiplier combinations across the selected bounds rather than only combinations expected to improve quality. So, the weaker combinations were also retained because they provided examples that the scoring models should assign lower predicted scores. Although Figure 17 presents three multipliers in separate panels, the geometry, appearance, and densification values were applied together as one complete combination in each exploration run. The panels should therefore not be interpreted as showing the independent effect of one multiplier.

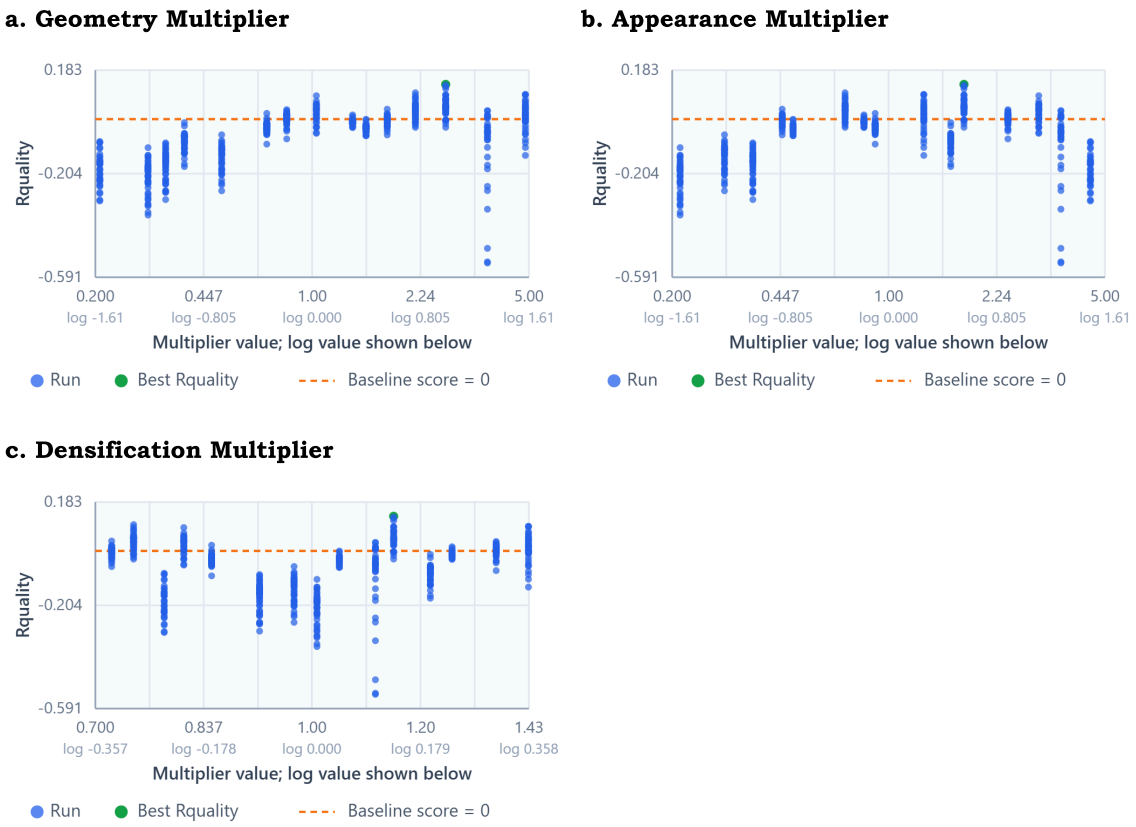


Figure 17 Relative quality scores across the tested geometry, appearance, and densification multiplier values. Each point represents one training row. The dashed orange line marks a relative quality score of zero, while the green point marks the highest score shown in each panel. The same relative quality score point is represented in all three charts.

A second version of the training data was prepared by adding the 37 hard-cap runs as penalty rows. It contained 690 rows, consisting of the same 653 normally completed exploration rows and 37 penalty rows. This version was used only as a supporting comparison because the models trained without the penalty rows performed better on the test projects. The comparison is presented in Section 6.4.3.

6.3 Scoring Model Training

The final dataset of 653 normally completed exploration rows was used to train the Ridge Regression and MLP scoring models. The model inputs, architecture, regularization, and training procedures are described in Section 4.3.

The Ridge Regression scoring model selected a regularization value $\lambda = 0.1$. The final model obtained a training RMSE of 0.0511. RMSE measures the typical difference between the predicted and observed relative quality scores while giving greater weight to larger prediction errors. Therefore, the value indicates that the predicted training scores differed from the observed scores by approximately 0.0511 on the relative quality score scale. As this value was calculated using the training data, it describes the model fit but does not by itself show how well the model generalizes to unseen projects.

The MLP was trained using a project-level split so that rows from the same project did not appear in both the training and validation sets. The split contained 36 training

projects and 10 validation projects, corresponding to 514 training rows and 139 validation rows. The best validation loss was 0.00282 at epoch 324. Training stopped at epoch 374 after the validation loss did not improve during the 50-epoch early-stopping period. The final training and validation losses were 0.00217 and 0.00287, respectively. The model configuration and early-stopping settings are described in Section 4.3.3.

The following table shows the five Ridge Regression terms with the largest coefficient magnitudes. The positive and negative coefficients are shown to indicate the direction of the learned relationships. The model terms, including the squared terms, descriptor–multiplier interaction terms, are defined in Section 4.3.2. All coefficient values are presented in **Appendix 2**.

Table 8 Ridge Regression terms with the largest coefficient magnitudes

Model term	**Coefficient**
Squared densification multiplier	+0.3449
Vegetation cover × geometry multiplier	−0.0682
Vegetation complexity × geometry multiplier	+0.0655
Geometry multiplier	+0.0427
Texture density × densification multiplier	−0.0373

The squared densification-multiplier term had the largest coefficient magnitude (+0.3449). This indicates that the predicted relative quality did not change at a constant rate across the tested densification range. The effect became stronger as the densification multiplier moved farther from the default value. The positive geometry-multiplier coefficient (+0.0427) also shows a general pattern in which higher geometry multipliers were linked to higher predicted scores when the other inputs remained unchanged. The descriptor–multiplier interaction terms indicate that this pattern varied between project conditions. With higher vegetation cover, increasing the geometry multiplier was associated with a smaller increase in the predicted score (−0.0682), while higher vegetation complexity was associated with a larger increase in the predicted score (+0.0655). Higher texture density was associated with a smaller increase in the predicted score when the densification multiplier increased (−0.0373). These coefficients show patterns learned from the available training data and should not be treated as evidence of direct cause and effect.

6.4 Evaluation on Test Projects

The final Ridge Regression and MLP scoring models were evaluated on the 12 unseen test projects described in Section 4.4. For each project, the model-selected run was compared with the corresponding baseline run. The project indices used in the following figures are listed on the table below.

Table 9 Test-project indices used in the result figures

Index	Project name	Index	Project name
Project 1	Maisonneuve Market	Project 7	Knínice Church of the Exaltation of the Holy Cross
Project 2	Thomas More Church	Project 8	Pix4D Forensic
Project 3	Brasov	Project 9	Telč St. James – 18 March
Project 4	Changunarayan Temple	Project 10	Angel Island State Park – Group 3
Project 5	Château Dívčí Hrad – Circular Flight	Project 11	Morice
Project 6	KTM20 Oblique – Flight 006	Project 12	Unčovice

6.4.1 Ridge Regression Results

The Ridge Regression model selected one multiplier combination for each of the 12 test projects. This section first presents the selected multiplier combinations and then compares the Ridge-selected runs with the corresponding baseline runs. The 30 candidate multipliers were passed to the Ridge regression model to predict the quality scores. Then among these predicted quality scores, the multiplier with highest score is selected as shown below.

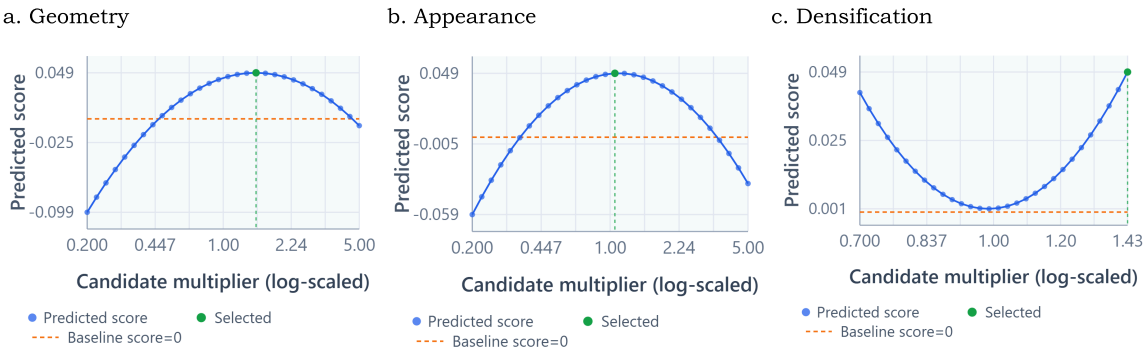


Figure 18 Predicted quality scores using the 30 candidate multipliers and descriptors as the model input for Maisonneuve Market (Project 1).

For this project, the selected geometry, appearance, and densification multipliers were 1.47, 1.06, and 1.43, respectively. The geometry and appearance scores reached their highest values near the middle of the tested ranges, while the selected densification value was near the upper bound. The three values were applied together as one multiplier combination.

The final Ridge-selected multiplier combinations for all 12 test projects are shown in the figure below.

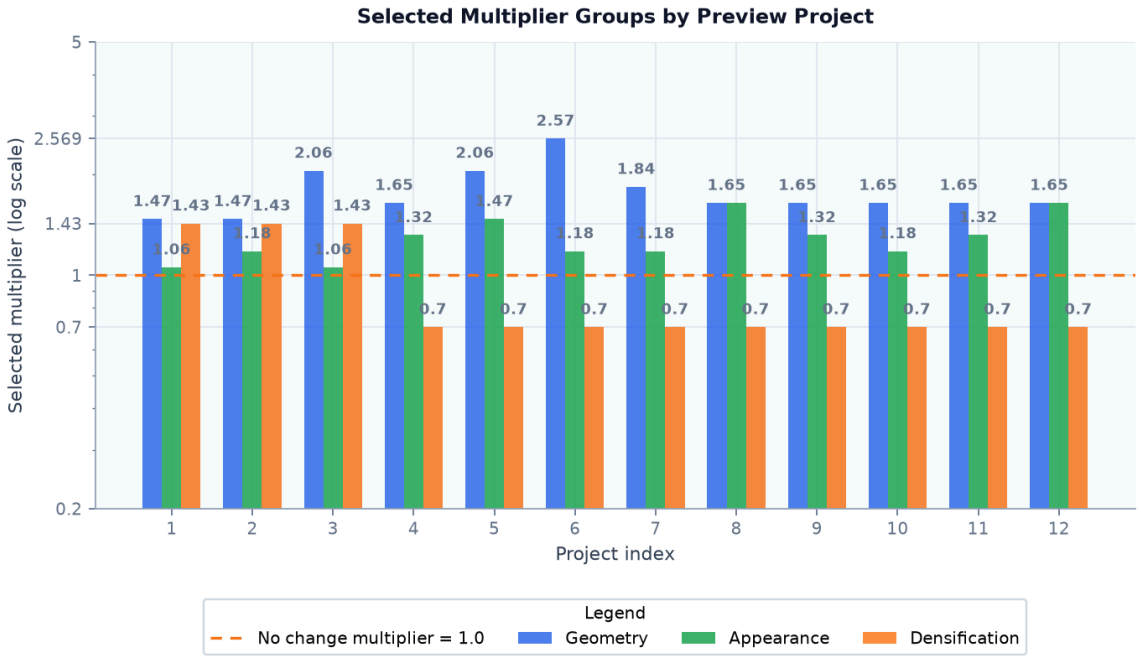
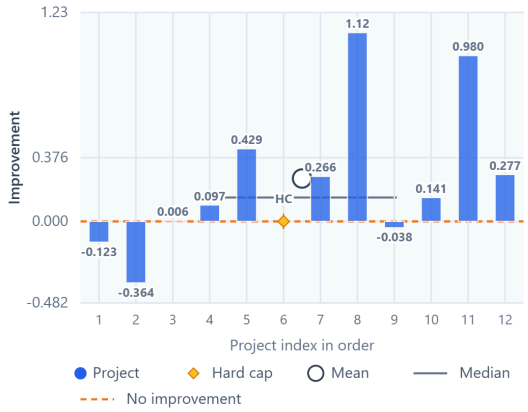


Figure 19 Geometry, appearance, and densification multipliers selected by Ridge Regression for the 12 test projects. The dashed line at 1.0 represents the default value.

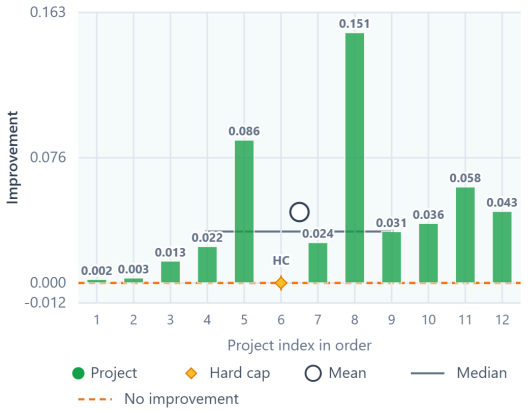
The selected combinations differed between projects. Geometry multipliers were above the default value for all projects. Appearance multipliers were also above the default for most projects, while the densification multiplier was reduced to 0.70 for most projects. The values represent complete combinations and should not be interpreted as separate effects of the three different multiplier groups.

The Ridge-selected run completed normally for 11 of the 12 test projects. The run for KTM20 Oblique – Flight 006 (Project 6) reached the Gaussian hard cap and did not produce final quality metrics. Among the 11 normally completed runs, the mean PSNR difference was +0.2536 dB, the mean SSIM difference was +0.0428, and the mean LPIPS improvement was +0.0835. All three metrics improved together in eight projects. Pix4D Forensic (Project 8) showed the strongest overall improvement, with gains in PSNR, SSIM, and LPIPS. Thomas More Church (Project 2) showed the weakest overall result, with a decrease in PSNR and only small improvements in SSIM and LPIPS. The complete project-level results are provided in **Appendix 3**.

a. PSNR



b. SSIM



c. LPIPS

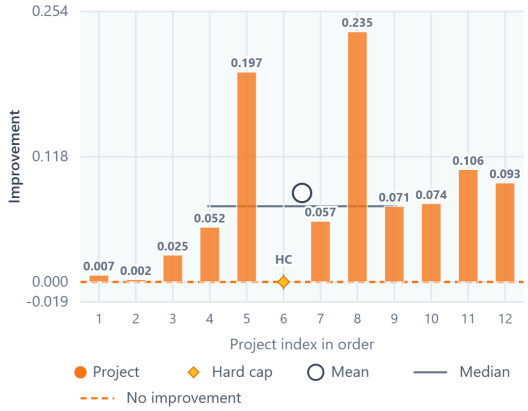


Figure 20 PSNR, SSIM, and LPIPS differences between the Ridge Regression model-selected and baseline runs for the 12 test projects. Positive values indicate improvement over the baseline. The horizontal line in each chart shows the median difference, while the circle marks the mean difference. KTM20 Oblique – Flight 006 (Project 6) is marked as Gaussian-hard-cap run and was excluded from the mean and median calculations. Project numbers correspond to the project order provided in Table 9.

Among the 11 normally completed Ridge runs, PSNR improved in eight projects, with a mean difference of +0.2536 dB and a median difference of +0.1413 dB. SSIM improved in all 11 projects, with a mean difference of +0.0428 and a median difference of +0.0311. LPIPS also improved in all 11 projects, with a mean improvement of +0.0835 and a median improvement of +0.0711. PSNR differences ranged from −0.3636 dB to +1.1160 dB, SSIM differences ranged from +0.0022 to +0.1513, and LPIPS improvements ranged from +0.0024 to +0.2351. As also summarized in Table 6, SSIM and LPIPS improved more consistently across the completed projects than PSNR.

Visual Assessment of Selected Ridge Results

Four Ridge results were selected for visual assessment to support the quantitative comparison. Pix4D Forensic (Project 8) and Château Dívčí Hrad – Circular Flight (Project 5) were selected because they showed clear improvements in PSNR, SSIM, and LPIPS. Thomas More Church (Project 2) and Maisonneuve Market (Project 1) were selected because their improvements were smaller or mixed across the three metrics. The higher and lower performing examples are presented separately below.

a. Pix4D Forensic (Project 8) Baseline



b. Pix4D Forensic (Project 8) model-selected



c. Château Dívčí Hrad – Circular Flight (Project 5) baseline



D. Château Dívčí Hrad – Circular Flight (Project 5) model-selected



Figure 21 Visual comparison of two higher-performing Ridge results at 5,000 iterations. The red boxes highlight areas where differences between the baseline and Ridge model-selected renderings are visible.

Pix4D Forensic showed the largest overall improvement. PSNR increased by 1.1160 dB, SSIM increased by 0.1513, and LPIPS improved by 0.2351. The red boxes show more continuous ground surfaces and clearer texture in the Ridge-selected result.

Château Dívčí Hrad – Circular Flight also improved in all three metrics. PSNR increased by 0.429 dB, SSIM increased by 0.086, and LPIPS improved by 0.197. The highlighted areas show clearer vegetation and ground details in the Ridge-selected result.

c. Thomas More Church (Project 2) Baseline



d. Thomas More Church (Project 2) model-selected



e. Maisonneuve Market (Project 1) baseline



F. Maisonneuve Market (Project 1) model-selected



Figure 22 Visual comparison of two lower-performing Ridge results at 5,000 iterations. The baseline and Ridge-selected renderings remain visually similar, so no specific areas are highlighted.

For Thomas More Church, PSNR decreased by 0.3636 dB, while SSIM improved by 0.0031 and LPIPS by 0.0024. The baseline and Ridge-selected renderings look very similar. The church structure, roof edges, windows, and surrounding scene are preserved, but no clear overall visual improvement can be seen in the Ridge-selected result.

For Maisonneuve Market, PSNR decreased by 0.123 dB, while SSIM improved by 0.002 and LPIPS by 0.007. The baseline and Ridge-selected renderings also look very similar. From this view, there is no clear visual improvement or noticeable loss of scene detail.

The visual assessment agrees with the quantitative results. The two higher-performing projects show clearer differences, while the lower-performing projects remain close to their baselines. These results therefore indicate limited improvement rather than major visual degradation. All comparisons were made at the fixed 5,000-iteration checkpoint and should be understood as reconstruction quality within the tested training budget, not as fully converged 3DGS quality.

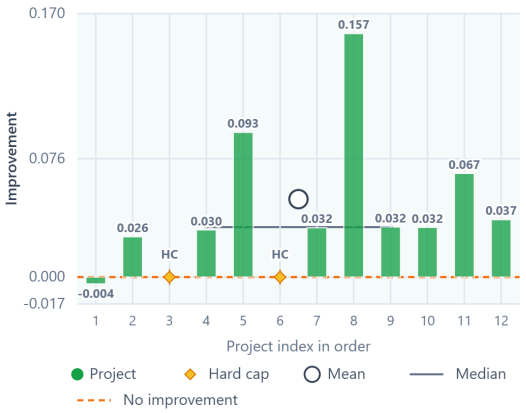
6.4.2 MLP Results

The MLP-selected runs completed normally in 10 of the 12 test projects. The runs for Brasov (Project 3) and KTM20 Oblique – Flight 006 (Project 6) reached the Gaussian hard cap and did not produce final quality metrics. Among the completed runs, Pix4D Forensic (Project 8) showed the strongest overall improvement, while Maisonneuve Market (Project 1) showed the weakest overall result. The complete project-level results are provided in **Appendix 3**.

a. PSNR



b. SSIM



c. LPIPS

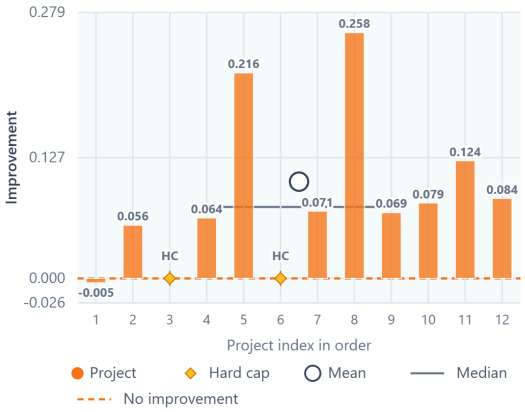


Figure 23 PSNR, SSIM, and LPIPS differences between the MLP model-selected and baseline runs for the 12 test projects. Positive values indicate improvement over the baseline. The horizontal line shows the median difference, while the circle marks the mean difference. Brasov (Project 3) and KTM20 Oblique – Flight 006 (Project 6) are marked as Gaussian-hard-cap runs and were excluded from the mean and median calculations. Project indices follow Table 9.

Among the 10 normally completed MLP runs, PSNR improved in six projects, with a mean difference of +0.2614 dB and a median difference of +0.2283 dB. SSIM improved in nine projects, with a mean difference of +0.0503 and a median difference of +0.0322. LPIPS also improved in nine projects, with a mean improvement of +0.1015 and a median improvement of +0.0750. PSNR differences ranged from −0.6786 dB to +1.1954 dB, SSIM differences ranged from −0.0044 to +0.1571, and LPIPS improvements ranged

from −0.0047 to +0.2579. As also summarized in Table 6, SSIM and LPIPS improved more consistently across the completed projects than PSNR.

6.4.3 Supporting Model Comparisons

A second version of the training data with 690 rows was prepared by adding the 37 Gaussian-hard-cap exploration runs as penalty rows. Each hard-cap run was assigned the lowest relative quality score observed among the normally completed exploration runs. The Ridge model trained with these penalty rows produced mean differences with the baseline of −0.0861 dB for PSNR, +0.0032 for SSIM, and +0.0074 for LPIPS improvement. The corresponding MLP model produced mean differences with the baseline of −0.1446 dB, +0.0158, and +0.0439, respectively. These results did not improve on the final models trained using only the 653 normally completed exploration rows. Therefore, the penalty-row models were kept only as supporting comparisons. The detailed comparison with main models is provided in **Appendix 3F**.

To check repeatability, the Ridge model-selected runs were repeated using the same multiplier combinations. Eleven of the 12 repeated runs completed normally. The repeated runs produced mean differences of +0.2120 dB for PSNR, +0.0417 for SSIM, and +0.0820 for LPIPS improvement. All three metrics improved together in eight projects. These results were close to the first Ridge evaluation, although the project-level values were not exactly the same. This shows that small run-to-run variation can occur even when the same project data, multiplier combinations, and training settings are used. The complete project-level values for this repeatability check are provided in **Appendix 3G**.

6.4.4 Ridge Results at the 7,000-Iteration Checkpoint

To evaluate the Ridge model-selected settings under a longer training budget and to check the suitability of the 5,000-iteration checkpoint, the baseline and Ridge model-selected configurations for all 12 test projects were rerun in a separate pipeline with a maximum of 7,000 iterations. The same trained Ridge model, COLMAP sparse reconstructions, selected multiplier combinations, Gaussian hard cap, and other processing settings were used. Only the maximum number of iterations was changed. Densification still ended at iteration 4,000, so the additional training from 5,000 to 7,000 iterations refined the existing Gaussian representation without creating new Gaussians. Each separate run was evaluated at both checkpoints. Therefore, these runs were not continuations of the main 5,000-iteration runs.

To check the overall quality changes at 7,000 iterations, the Ridge model-selected runs were compared with their corresponding baselines. The mean PSNR difference was +0.089 dB and the median was +0.084 dB. The mean SSIM difference was +0.034, with a median of +0.027. The mean LPIPS improvement was +0.073, with a median of +0.065. PSNR improved in seven of the 12 projects, SSIM improved in 11 projects, and LPIPS improved in all 12 projects.

To compare the 5,000- and 7,000-iteration checkpoints, the direction of the Ridge model-selected–baseline difference was examined for each project. The PSNR difference remained in the same direction for 11 projects, the SSIM difference for all 12 projects, and the LPIPS difference for 11 projects. The exact metric differences changed after

further training, but the overall result remained positive. This supports the use of 5,000 iterations as a practical fixed checkpoint for the main controlled experiment, although it should not be interpreted as full convergence. Complete project-level values are provided in **Appendix 3E**.

Table 10 Comparison of Ridge results at 5,000 and 7,000 iterations

Metric	Mean at 5,000	Median at 5,000	Mean at 7,000	Median at 7,000	Same direction
PSNR difference	+0.081 dB	+0.164 dB	+0.089 dB	+0.084 dB	11/12
SSIM difference	+0.029	+0.023	+0.034	+0.027	12/12
LPIPS improvement	+0.064	+0.053	+0.073	+0.065	11/12

The PSNR and SSIM differences were calculated as the Ridge-selected value minus the baseline value. LPIPS improvement was calculated as the baseline value minus the Ridge-selected value. Positive values mean that the Ridge-selected run performed better than the baseline. “Same direction” shows how many projects either improved at both checkpoints or performed worse at both checkpoints.

6.5 Outputs

In addition to the model-evaluation results, the research produced several practical and dissemination outputs. These outputs include the 3DGS processing and recommendation platform, an interactive platform for presenting selected test results, the thesis website, and the thesis poster. The processing platform supports the complete workflow. The visualization platform and thesis website provide simpler ways to explore and present the research results.

6.5.1 3DGS Processing and Recommendation Platform

The main technical output of this research is Bimba3D-re, the open-source platform described in Chapter 5. It supports project preparation, 3DGS processing, hyperparameter recommendation, test-run execution, and result comparison. The platform combines the processing workflow with the trained Ridge Regression and MLP scoring models. For a new project, it calculates the required descriptors, generates candidate multiplier combinations, predicts their relative quality scores, and selects one combination for testing.

The platform also records baseline and model-selected results, including PSNR, SSIM, LPIPS, runtime, Gaussian count, and evaluation images. A comparison page allows the baseline and model-selected runs to be examined together using metric tables, charts, image previews, and 3D Gaussian-splat visualizations.

The complete processing workflow requires substantial computational resources and is intended for local or server-based execution. Therefore, the full processing platform is provided through its source-code repository rather than being hosted as a public online

service. Anyone can pull the code and install it on their own computer, provided the hardware configurations are sufficient as described in Chapter 5.

6.5.2 Interactive Results Visualization Platform

A separate web-based results visualization platform was developed to make selected research outputs accessible without running the complete processing workflow. Unlike the main Bimba3D-re processing platform, it does not perform SfM reconstruction, 3DGS training, or model prediction. Instead, it presents prepared outputs from selected unseen test projects.

The platform includes examples of higher-performing and lower-performing model-selected results from the final evaluation. Users can open the selected 3D Gaussian splat output for each case, compare baseline and model-selected image previews using swipe views, view checkpoint-based PSNR, SSIM, LPIPS, and loss charts, and examine project-level metrics, selected multipliers, runtime, Gaussian count, and processing setup information. This provides an interactive way to explore the results reported in this chapter.

6.5.3 Thesis Website and Poster

A thesis website was created to provide a public overview of the research. It presents the research problem, objectives, methodology, model-development workflow, main results, and conclusions. It also includes links to the related software and research-output repositories, together with access to the interactive results platform and demonstration materials.

A thesis poster was also prepared, and submitted in printed form. It presents the research motivation, workflow, data, scoring models, test results, main conclusions, and developed platforms in a compact visual format.

7 DISCUSSION

This chapter interprets the results presented in Chapter 6. It discusses what the model results mean, how reconstruction quality was related to processing cost, and how the dataset size, project split, tested parameter ranges, and evaluation setup influenced the results. It also considers alternative approaches, future improvements, and the possible use of Copernicus data.

7.1 Interpretation of the Model Results

The results show that information from previous drone-image projects can support hyperparameter selection for unseen projects. Both scoring models selected multiplier combinations that improved the mean PSNR, SSIM, and LPIPS results among the normally completed runs. However, the two models showed different levels of reliability. Ridge Regression completed 11 of the 12 projects normally, while the MLP completed 10. Ridge improved SSIM and LPIPS in all 11 normally completed projects, whereas the MLP improved these metrics in nine of its 10 normally completed projects. All three metrics improved together in eight Ridge projects and six MLP projects.

The MLP was included as a comparison model to examine whether nonlinear relationships could improve multiplier selection. It produced slightly higher mean and median improvements than Ridge Regression when only the normally completed runs were considered. However, these values exclude the two MLP runs that reached the Gaussian hard cap. The averages therefore describe MLP performance only when training completed successfully and should not be interpreted as showing that the MLP was better overall. Ridge Regression completed one additional project, produced fewer hard-cap cases, and gave more consistent project-level improvements. Under the tested conditions, the results did not provide clear evidence that modelling nonlinear relationships improved recommendation reliability. Ridge Regression was more reliable overall, while the MLP produced stronger improvements in some cases but less consistent results. One possible reason is that the MLP learned more complex patterns, but the 46 independent training projects may not have provided enough variation for these patterns to work consistently on unseen projects.

Most multiplier combinations in the training data did not perform better than the default settings. Among the 653 training rows, 191 had a positive relative quality score and 462 had a negative score. This suggests that the default settings already provided a fairly strong starting point for many projects. Only some multiplier combinations improved quality, while many produced lower scores. So, the aim of the scoring models was not to find one combination that worked best for every project. Instead, the models learned to rank the candidate combinations differently based on the descriptors of each project.

Both models improved SSIM and LPIPS more consistently than PSNR. This may partly result from how the learning target was defined. The relative quality score combined normalized PSNR, SSIM, and LPIPS with equal weights. The models were therefore trained to find combinations that performed well across all three metrics, rather than combinations that maximized PSNR alone. A multiplier combination could still receive a

positive quality score when improvements in SSIM and LPIPS were large enough to balance a small decrease in PSNR.

The three metrics describe different parts of image quality. PSNR measures pixel-level error, while SSIM considers luminance, contrast, and local image structure (Wang et al., 2004). LPIPS compares deep image features and represents perceptual similarity (Zhang et al., 2018). Because the metrics focus on different image properties, they do not always change in the same direction. In some projects, PSNR decreased while SSIM and LPIPS improved. Thomas More Church is one example. Its Ridge-selected result remained visually close to the baseline, while PSNR decreased and the other two metrics improved only slightly. This shows why the reconstruction results should be assessed using all three metrics.

Differences between projects may also be related to image scale and viewing conditions. Yu et al. (2024) showed that changes in image scale can cause visual artifacts in standard 3DGS. Their Mip-Splatting method reduced these problems using scale-aware filtering. This helps explain why the same 3DGS settings may not work equally well for projects captured under different conditions.

The project-level results suggest some possible relationships between the project conditions and model performance. The strongest Ridge results, including Pix4D Forensic, Château Dívčí Hrad – Circular Flight, and Morice, generally had lower terrain-roughness values than the weakest cases. Some of these projects also contained relatively small image sets. However, the test set was too small to establish general relationships between individual descriptors and model performance. So, these observations should be treated as descriptive patterns in the small test set, rather than as general relationships that apply to all new projects.

The multiplier selections also require careful interpretation. Ridge Regression selected geometry and appearance multipliers above the default for all 12 test projects. However, it selected the lower densification bound for nine projects and the upper bound for three. The MLP selected the lower densification bound for all 12 projects. This repeated selection of boundary values is consistent with the Ridge model's strong squared densification term, which favoured values farther from the default. This repeated selection of boundary values suggests that available data and candidate combinations did not clearly represent the relationship between densification and the predicted score.

A lower densification multiplier reduced both the densification-gradient threshold and the opacity-removal threshold. Under the implementation used in this study, this could allow more Gaussians to be created or retained. Most of the larger quality improvements were also associated with an increase in Gaussian count. However, the three multiplier values were always applied together, and different projects received different geometry and appearance values. The results therefore do not prove that the densification multiplier alone caused either the additional Gaussians or the quality improvements. Users should therefore apply changes to densification-related settings carefully, especially for projects that already produce a large number of Gaussians with the default settings. Gaussian growth should be monitored during training, and a suitable hard cap should be set according to the available GPU memory to avoid excessive processing time or incomplete runs.

The Ridge coefficients also need to be interpreted carefully. The squared densification term had the largest coefficient in absolute value. This indicates that its relationship with the predicted score was not a simple constant increase or decrease. Several of the larger coefficients were also interactions between project descriptors and multipliers. This is consistent with the main idea of the thesis that useful multiplier values may change with project conditions. However, these coefficients only describe patterns learned from the available data. They should not be interpreted as direct causal effects.

Fang et al. (2025) also showed that changing selected 3DGS settings can improve reconstruction quality. In their indoor-scene experiment, they tested settings such as image resolution, the spatial-position gradient threshold, and the scaling learning rate. Their selected configuration increased PSNR by 4.3%, increased SSIM by 0.2%, and reduced LPIPS. Their study tested parameter values directly on one indoor scene. In contrast, this thesis learned multiplier recommendations from several drone-image projects. The reported improvements are therefore not directly comparable. However, both studies indicate that the default settings may not be equally suitable for every project.

Li et al. (2025) reported similar benefits from adaptive hyperparameter tuning in RLGS. On the Tanks and Temples dataset, their method increased PSNR from 24.04 to 24.74 dB and SSIM from 0.851 to 0.866. LPIPS decreased from 0.170 to 0.158. RLGS adjusts selected settings during training, while the method developed in this thesis selects one multiplier combination before training starts. The results cannot therefore be compared directly. Still, both approaches show that changing 3DGS settings can improve reconstruction quality. Li et al. achieved their improvements while keeping the number of Gaussians within a fixed limit. This shows that better quality does not always require more Gaussians. However, this thesis showed that quality improvements were often linked to higher Gaussian counts and longer processing times. Adding Gaussian count and runtime to the learning target could therefore support a better balance between quality and processing cost.

7.2 Quality, Processing Cost and Run Stability

The main Ridge and MLP comparison was conducted at the fixed 5,000-iteration checkpoint. The additional Ridge evaluation at 7,000 iterations showed that absolute metric values continued to change, but the overall Ridge–baseline result remained similar. At 7,000 iterations, the Ridge-selected settings still produced positive mean and median differences for PSNR, SSIM, and LPIPS. The direction of the project-level comparison also remained unchanged in most cases. This supports the use of 5,000 iterations as a practical checkpoint for the large number of controlled runs. The extended evaluation included only the baseline and Ridge-selected runs, so the comparison between Ridge Regression and MLP remains based on the 5,000-iteration results.

The fixed checkpoint was necessary because the study required hundreds of complete 3DGS runs. Increasing the number of iterations for every baseline, exploration, and model-selected run would have substantially increased the processing time. The 1,400-pixel image limit and the Gaussian hard cap were practical processing limits selected for the available 8 GB GPU. These choices allowed the projects to be compared

under consistent conditions, but they limit how far the results can be transferred to longer training, higher-resolution images, or more powerful hardware.

The selected settings did not reduce processing cost. Among the normally completed runs, Ridge Regression produced approximately 953,000 additional Gaussians and required about 167 additional seconds on average. The MLP produced approximately 1.41 million additional Gaussians and required about 222 additional seconds on average. So, MLP achieved slightly higher mean quality changes among its successful runs, but at a higher average processing cost and with one additional hard-cap case.

Other studies have tried to improve quality without increasing processing cost. For example, LightGaussian reduced model size by about 15 times and increased rendering speed from 144 to 237 frames per second while keeping similar visual quality (Fan et al., 2024). The workflow in this thesis focused only on reconstruction quality, not on output 3D model size or runtime. This helps explain why the recommended settings often produced more Gaussians and required more processing time.

This distinction is important for the meaning of “optimization” in the thesis. The workflow reduced the need for manual hyperparameter trial-and-error and selected settings using previous data. It did not optimize runtime, memory use, or model size. Its learning target represented image quality only. The result is therefore a quality-oriented recommendation rather than a complete quality–efficiency optimization.

The hard-cap cases also provide useful information. In the main 5,000-iteration evaluation, KTM20 Oblique – Flight 006 reached the Gaussian hard cap for both Ridge and MLP, while Brasov reached it only for the MLP. Both projects already had relatively high Gaussian counts in their baseline runs, and both MLP-selected runs used the lower densification multiplier. This suggests that baseline Gaussian count and early training behaviour may help identify hard-cap risk. These factors were not included in the current quality-scoring target. In the separate 7,000-iteration Ridge pipeline, KTM20 completed normally using the same SfM data, selected multipliers, and Gaussian hard cap. This shows that Gaussian growth can vary between separate 3DGS training runs.

The Ridge-selected multiplier combinations were tested a second time to examine repeatability. The repeated runs produced similar mean SSIM and LPIPS improvements, the same number of projects with all three metrics improved, and the same hard-cap project. However, the project-level values were not identical, and PSNR showed more variation than SSIM and LPIPS. This indicates that a single 3DGS run is not perfectly deterministic even when the images, SfM outputs, hyperparameters, and main processing settings remain unchanged. The overall Ridge findings were similar in both runs, but the exact metric improvement for an individual project may change slightly when the same run is repeated. Hence, the future evaluations should repeat the baseline and model-selected runs several times for a representative group of projects and report the mean and standard deviation of the results.

Hardware stability was also a practical concern. A four-minute cooldown period was added between consecutive runs to limit laptop overheating during the nine-day offline experiment used to generate the training rows. This increased the total processing time. A workstation or server with better cooling and more GPU memory would reduce this

limitation and make it possible to test higher-resolution images, larger Gaussian limits, and longer training checkpoints.

7.3 Limitations

The main data limitation was the number of distinct projects. Although the final training data contained 653 rows, these rows came from only 46 model-development projects. Rows from the same project shared the same scene descriptors and were therefore not fully independent. The effective variety of the dataset was mainly determined by the 46 projects rather than by the total number of rows.

Only 15 exploration runs were completed for each project. This limited the coverage of the multiplier combinations and may have left useful combinations untested. More exploration runs could provide a clearer pattern, although they would require additional processing time.

The tested multiplier ranges were limited to reduce unstable runs and excessive Gaussian growth. As a result, potentially useful values outside these ranges were not evaluated. Wider ranges could be tested in future experiments with suitable limits on memory use and Gaussian count.

The project descriptors may not fully describe reconstruction difficulty. Additional information such as sparse-point density, reprojection error, lighting conditions, moving objects, reflective surfaces, and repeated textures could help explain differences between projects. However, adding many descriptors without increasing the number of independent projects could also increase overfitting.

The models were trained using the descriptor ranges represented by the 46 model-development projects. Recommendations may therefore be less reliable when a new project has descriptor values that are very different from the training projects. Future versions could check whether a new project falls outside the ranges seen during training and warn the user when the recommendation may be less reliable.

The final evaluation contained 12 unseen projects. This allowed an initial comparison, but it was not enough to represent all scene types, flight patterns, cameras, and environmental conditions. The findings are therefore limited to the types of projects represented in this study. A larger external test set would provide stronger evidence that the models can work on other projects.

7.4 Alternative Approaches and Future Development

One alternative would be to optimize the hyperparameters separately for every new project using random search, Bayesian optimization, or an adaptive resource-allocation method. Such methods could search more directly for the best setting for one project. However, they would require several new 3DGS runs for a single project. The current offline approach reduces this deployment-time search by reusing information collected from earlier projects. Its limitation is that the recommendation depends on how well the new project is represented by the training data.

Another option is to adjust hyperparameters during training. RLGS changes selected settings while a reconstruction is running, which allows it to respond to the current training behaviour. In comparison, an offline recommendation is easier to add to an existing workflow because it does not require a control policy inside the training loop. A combined method may be more effective. The offline model could provide the starting settings, while a simpler online method could later adjust densification or learning rates based on loss, Gaussian growth, and validation behaviour.

The Ridge regularization strength was selected using MSE calculated from the training data. A separate project-level validation set was not used for this step. Therefore, the selected value mainly represents how well the model fitted the available training rows. It may not be the value that generalizes best to unseen projects. Future studies should select the regularization strength through project-level cross-validation or with a separate validation group.

Future model development should first include more independent projects from a wider range of conditions. With a larger dataset, models such as Random Forest, gradient boosting, Gaussian Process Regression, and Support Vector Regression could be tested using the same project-level validation procedure. Runs from the same project should remain within the same data group during validation. The models should then be compared using final test results and run stability, rather than only training or validation error.

The recommendation target should also be extended beyond quality. A multi-objective model could balance PSNR, SSIM, LPIPS, runtime, Gaussian count, memory use, and hard-cap risk. The user could then choose between a quality-focused result and a more computationally efficient result. For example, a small quality improvement with moderate Gaussian growth may be preferable to a slightly larger improvement requiring several million additional Gaussians. Future work could also test and recommend the selected hyperparameters individually instead of using one multiplier for each parameter group. This would allow more flexible settings but would require more projects and exploration runs.

7.5 Potential Use of Copernicus Data in 3DGS

Copernicus data were not used in the experiments of this thesis. Their main value for future 3DGS work would be at a wider geographic scale. Drone images provide fine local detail, but they usually cover only the planned flight area. Copernicus products could provide information about the surrounding terrain, land cover, vegetation, water, and built-up environment. The Copernicus Land Monitoring Service provides products describing land cover, land use, vegetation, forests, croplands, water, wetlands, and impervious surfaces (Copernicus Land Monitoring Service, n.d.-a; Copernicus Land Monitoring Service, n.d.-b). These products could help place a drone-based 3DGS reconstruction within its wider geographic setting and support applications that extend beyond the flight boundary.

Sentinel-2 provides multispectral imagery at 10 m, 20 m, and 60 m spatial resolutions (Copernicus Data Space Ecosystem, n.d.-d). These data are useful for describing broader environmental conditions, but their resolution limits their use for detailed

reconstruction of individual buildings and small objects. However, Copernicus Contributing Missions provide additional Earth-observation data from optical systems such as WorldView, GeoEye, and Pléiades (Copernicus Data Space Ecosystem, n.d.-a). These collections supplement the Sentinel missions and include very-high-resolution imagery that may support satellite-based reconstruction when several overlapping views and suitable camera information are available. Access to these products depends on the specific dataset and their Copernicus Contributing Missions user category (Copernicus Data Space Ecosystem, n.d.-b).

Recent studies show how 3DGS can be adapted to multi-view satellite imagery, but this does not apply equally to all Copernicus data. Sentinel-2 provides repeated observations of the same area over time, but these images are generally not acquired from the multiple viewing directions needed for detailed 3DGS reconstruction. In contrast, some very-high-resolution Copernicus Contributing Mission datasets may include suitable overlapping views and camera information. Aira et al. (2025) demonstrated this using several WorldView-3 images of the same locations, acquired from different viewing directions and on different dates. Their method adapted Gaussian Splatting to satellite camera models and reduced appearance differences caused by changes in brightness, colour, and shadows between acquisition dates. The study therefore shows that satellite-based 3DGS is possible when suitable multi-view imagery and camera information are available, but this is more relevant to selected Copernicus Contributing Missions than to standard Sentinel-2 data.

A second direction is to combine a locally detailed drone reconstruction with satellite information covering a wider area. Teppati Losè et al. (2021) used very-high-resolution satellite imagery and UAV photogrammetry to document areas affected by the October 2020 flood in Limone Piemonte, Italy. Their study followed a multi-sensor and multi-scale approach. Satellite imagery supported mapping across the larger affected area, while UAV data provided more detailed documentation of selected locations (Teppati Losè et al., 2021). A similar future workflow could combine Copernicus Emergency Management Service maps with drone-based 3DGS models. The Copernicus Emergency Management Service provides information on event extent, such as flooded or burned areas, and on damage to buildings, transport networks, and other infrastructure (Copernicus Emergency Management Service, 2025). These products could show the regional extent of an event, while 3DGS could provide a realistic and interactive view of a damaged building, bridge, road, river section, or settlement. The main role of the 3DGS model would be to make local conditions easier to inspect and communicate visually.

Forest monitoring provides another example of how Copernicus and drone data can serve different roles. Ács et al. (2026) combined Sentinel-2 vegetation-change analysis with high-resolution UAV photogrammetry for forest-damage assessment in the Sopron Mountains. Sentinel-2 data were used to identify potentially damaged forest patches, while the UAV survey supported detailed assessment at the individual-tree level and validation of the satellite results (Ács et al., 2026). A similar future workflow could use Sentinel-2 to identify areas of vegetation change and drone-based 3DGS to provide a visual representation of selected forest stands. The 3DGS model could provide an interactive visual record of the local scene, while Sentinel-2 supplies repeated information over the wider forest area.

Cultural heritage is another application in which the visual strengths of 3DGS are useful. Yu et al. (2025) used UAV images and 3DGS to create detailed visualizations of modern architectural heritage. Their study developed a workflow for UAV-based heritage documentation and 3D visualization using the Delft University of Technology Aula and Library buildings as case studies (Yu et al., 2025). Copernicus products could extend this type of local heritage reconstruction by adding information about the surrounding terrain, vegetation, water, land cover, and urban environment. For example, a 3DGS model of a heritage site could be displayed within its wider environmental setting. This could support virtual visits, education, tourism, and public communication by helping users understand both the appearance of the site and its relationship with the surrounding landscape.

Copernicus data could also complement some of the project descriptors used in this thesis. Land-cover products can describe broad environmental conditions such as forests, croplands, water, wetlands, grasslands, and built-up areas, while Sentinel-2 and the Copernicus DEM could provide general vegetation and terrain information (Copernicus Land Monitoring Service, n.d.-b; Copernicus Data Space Ecosystem, n.d.-c; Copernicus Data Space Ecosystem, n.d.-d). Future research could test whether these sources can provide a limited set of environmental descriptors for older projects when the original drone images are not available. If reconstruction results and tested hyperparameter settings are still available, such projects might contribute partial information to a larger training dataset. However, Copernicus data cannot recover image-specific descriptors such as camera angle, overlap, heading consistency, texture detail, or motion blur. They should therefore be treated as supplementary inputs, and their use with incomplete descriptor sets would need careful evaluation.

Overall, Copernicus data should be considered complementary to drone-based 3DGS rather than a replacement for detailed local imagery. Sentinel-2, land-cover products, and elevation data could provide wider environmental context, while selected Copernicus Contributing Mission datasets may support direct satellite-based 3DGS when suitable multi-view imagery and camera information are available. In combined applications, Copernicus products could describe the wider area, while 3DGS provides a detailed, interactive, and photorealistic view of selected locations. Future research could examine how these data sources can be integrated for environmental monitoring, disaster communication, cultural heritage, and other geographically referenced visual applications.

8 CONCLUSION

The main goal of this thesis was to design and evaluate an AI-guided workflow for selecting 3DGS hyperparameters for drone-image projects. The workflow used information from previous reconstruction projects to recommend suitable starting settings for new projects. The study examined whether these recommendations could improve or maintain reconstruction quality compared with the default 3DGS settings. It also aimed to provide a learned starting point for projects where reconstruction quality is important and testing many parameter combinations would be costly.

The study used 58 drone-image projects representing different scenes, acquisition patterns, image characteristics, and geographic locations. Of these, 46 projects were used for model development, while 12 projects were kept separate for final testing. Baseline runs were first completed using the default settings. Controlled exploration runs were then carried out using different geometry, appearance, and densification multiplier combinations.

The model-development stage produced 653 usable training rows from normally completed exploration runs. Each row connected the descriptors of one project, a tested multiplier combination, and the relative quality obtained in comparison with the project-specific baseline. These rows were used to train Ridge Regression as the main scoring model and a compact multilayer perceptron as a nonlinear comparison model.

For each unseen test project, the trained models evaluated 30 candidate multiplier combinations and selected the combination with the highest predicted relative quality score. The selected settings were then applied in an actual 3DGS run and compared with the corresponding baseline at the 5,000-iteration checkpoint using PSNR, SSIM, LPIPS, runtime, Gaussian count, and visual results.

The final Ridge Regression model completed 11 of the 12 test projects normally. Among these completed runs, PSNR improved in eight projects, while SSIM and LPIPS improved in all 11 projects. All three metrics improved together in eight projects. The mean differences from the baseline were +0.254 dB for PSNR, +0.043 for SSIM, and +0.084 for LPIPS improvement. One Ridge-selected run reached the Gaussian hard cap and did not produce complete final quality metrics. These results show that the model generally selected useful starting settings for unseen projects, although improvement was not achieved for every metric in every project. A separate evaluation of the baseline and Ridge model-selected runs at 7,000 iterations produced positive mean and median differences for all three quality metrics and broadly supported the result observed at the 5,000-iteration checkpoint.

The MLP also produced positive mean improvements among its normally completed runs. However, it completed only 10 of the 12 projects normally, compared with 11 for Ridge Regression. Its project-level results were also less consistent. Ridge Regression was therefore considered the more reliable model under the conditions of this study. The result supports the use of a simpler regularized model when the number of independent training projects is limited.

The recommended settings generally increased the number of Gaussians and processing time compared with the baseline. The workflow therefore improved

reconstruction quality within the fixed training budget, but it did not reduce computational cost. In this thesis, optimization mainly referred to learning-guided hyperparameter selection and reduced manual trial-and-error rather than faster or smaller reconstruction.

The research objectives were achieved. A consistent baseline workflow was prepared, controlled exploration data were created, scoring models were developed, and the selected multiplier combinations were evaluated on projects that were not used during model development. The findings show that previous 3DGS reconstruction projects can provide useful information for selecting hyperparameters for new drone-image projects. The recommendations should nevertheless be treated as quality-oriented starting settings rather than guaranteed optimal values.

The main contributions of the thesis are the structured training data produced from controlled 3DGS experiments, the learning-based method for scoring and selecting multiplier combinations, and the implementation of the complete workflow in the open-source Bimba3D-re platform. The platform supports project preparation, SfM processing, 3DGS reconstruction, training-data preparation, scoring-model training, model testing, and result comparison. An interactive results platform, thesis website, and poster were also prepared to present and communicate the research outputs.

Future work should use a larger and more diverse collection of independent projects. Increasing the number of exploration runs would provide broader coverage of the multiplier space and allow other learning models to be evaluated more reliably. The recommendation process could also consider reconstruction quality together with runtime, Gaussian count, memory use, image resolution, and training length. A separate future research project could also examine how Copernicus satellite images can support wider-area 3DGS reconstruction and visualization.

Overall, the thesis demonstrates that knowledge collected from previous 3DGS reconstruction runs can support hyperparameter selection for unseen drone-image projects. The workflow is most useful when reconstruction quality is important and testing many parameter combinations for every new project would require too much time and computation.

REFERENCES AND INFORMATION SOURCES

ÁCS, N., HEIL, B., SZÁSZ, B., FOLCZ, Á., PREISINGER, M., SÁNDOR, G., CZIMBER, K. (2026): Two-phase forest damage assessment with Sentinel-2 NDVI double differencing and UAV-based segmentation in the Sopron Mountains. Remote Sensing, 18(5), 803. doi:10.3390/rs18050803.

AIRA, L. S., FACCIOLO, G., EHRET, T. (2025): Gaussian Splatting for efficient satellite image photogrammetry. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5959–5969. doi:10.1109/CVPR52734.2025.00559.

BERGSTRA, J., BENGIO, Y. (2012): Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13(10), pp. 281–305.

BISCHL, B., BINDER, M., LANG, M., PIELOK, T., RICHTER, J., COORS, S., THOMAS, J., ULLMANN, T., BECKER, M., BOULESTEIX, A.-L., DENG, D., LINDAUER, M. (2023): Hyperparameter optimization: Foundations, algorithms, best practices and open challenges. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 13(2), e1484. doi:10.1002/widm.1484.

BORISOV, V., LEEMANN, T., SEßLER, K., HAUG, J., PAWELCZYK, M., KASNECI, G. (2024): Deep neural networks and tabular data: A survey. IEEE Transactions on Neural Networks and Learning Systems, 35(6), pp. 7499–7519. doi:10.1109/TNNLS.2022.3229161.

BREIMAN, L. (2001): Random forests. Machine Learning, 45(1), pp. 5–32. doi:10.1023/A:1010933404324.

CHAIBUB NETO, E., BARE, J. C., MARGOLIN, A. A. (2014): Simulation studies as designed experiments: The comparison of penalized regression models in the “large p, small n” setting. PLoS ONE, 9(10), e107957. doi:10.1371/journal.pone.0107957.

COPERNICUS DATA SPACE ECOSYSTEM [online] (n.d.-a): Copernicus Contributing Missions. [cit. 2026-07-29]. Available online: https://dataspace.copernicus.eu/explore-data/data-collections/copernicus-contributing-missions

COPERNICUS DATA SPACE ECOSYSTEM [online] (n.d.-b): Copernicus Contributing Missions User Categories. [cit. 2026-07-29]. Available online:

https://dataspace.copernicus.eu/explore-data/data-collections/copernicus-contributing-missions/ccm-user-categories

COPERNICUS DATA SPACE ECOSYSTEM [online] (n.d.-c): Digital Elevation Model (DEM) Data. [cit. 2026-07-29]. Available online: https://documentation.dataspace.copernicus.eu/APIs/SentinelHub/Data/DEM.html

COPERNICUS DATA SPACE ECOSYSTEM [online] (n.d.-d): Sentinel-2. [cit. 2026-07-29]. Available online: https://documentation.dataspace.copernicus.eu/Data/SentinelMissions/Sentinel2.html

COPERNICUS EMERGENCY MANAGEMENT SERVICE [online] (2025): About Copernicus EMS On-Demand Mapping. [cit. 2026-07-29]. Available online: https://mapping.emergency.copernicus.eu/about/

COPERNICUS LAND MONITORING SERVICE [online] (n.d.-a): About. [cit. 2026-07-29]. Available online: https://land.copernicus.eu/en/about?tab=what_we_offer

COPERNICUS LAND MONITORING SERVICE [online] (n.d.-b): Portfolio overview. [cit. 2026-07-29]. Available online: https://land.copernicus.eu/en/products?tab=full_coverage_land_cover__use

ESRI [online] (n.d.): Air Traffic Control; Red Rocks Amphitheatre; Angel Island; Trakai Island Castle. ESRI sample drone datasets. [cit. 2026-06-18]. Available online: https://www.esri.com/en-us/arcgis/products/arcgis-reality/resources/sample-drone-datasets.

FAN, Z., WANG, K., WEN, K., ZHU, Z., XU, D., WANG, Z. (2024): LightGaussian: Unbounded 3D Gaussian compression with 15x reduction and 200+ FPS. In: Advances in Neural Information Processing Systems 37, pp. 140138–140158. doi:10.52202/079017-4447.

FANG, X., ZHANG, Y., TAN, H., LIU, C., YANG, X. (2025): Performance evaluation and optimization of 3D Gaussian Splatting in indoor scene generation and rendering. ISPRS International Journal of Geo-Information, 14(1), 21. doi:10.3390/ijgi14010021.

FERNANDES, S., PRAKASH, A., GRANDERSON, J. (2023): Unmanned Aerial Vehicle Image Dataset of the Built Environment for 3D Reconstruction (UAVID3D) [dataset]. Zenodo repository. doi:10.5281/zenodo.7968619.

FEURER, M., KLEIN, A., EGGENSPERGER, K., SPRINGENBERG, J. T., BLUM, M., HUTTER, F. (2015): Efficient and robust automated machine learning. In: Advances in Neural Information Processing Systems 28, pp. 2962–2970.

FRIDOVICH-KEIL, S., YU, A., TANCIK, M., CHEN, Q., RECHT, B., KANAZAWA, A. (2022): Plenoxels: Radiance fields without neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5501–5510. doi:10.1109/CVPR52688.2022.00542.

GRAPHDECO-INRIA [online] (n.d.): 3D Gaussian Splatting for Real-Time Radiance Field Rendering: Official reference implementation. [cit. 2026-07-29]. Available online: https://github.com/graphdeco-inria/gaussian-splatting

HOERL, A. E., KENNARD, R. W. (1970): Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 12(1), pp. 55–67. doi:10.1080/00401706.1970.10488634.

JÄGER, M., HILLEMANN, M., JUTZI, B. (2025): FeatureGS: Eigenvalue-feature optimization in 3D Gaussian Splatting for geometrically accurate and artifact-reduced reconstruction. ISPRS Open Journal of Photogrammetry and Remote Sensing, 17, 100100. doi:10.1016/j.ophoto.2025.100100.

KERBL, B., KOPANAS, G., LEIMKÜHLER, T., DRETTAKIS, G. (2023): 3D Gaussian Splatting for real-time radiance field rendering. ACM Transactions on Graphics, 42(4), Article 139. doi:10.1145/3592433.

LI, L., JAMIESON, K., DESALVO, G., ROSTAMIZADEH, A., TALWALKAR, A. (2018): Hyperband: A novel bandit-based approach to hyperparameter optimization. Journal of Machine Learning Research, 18(185), pp. 1–52.

LI, Z., ZHAN, H., LI, C., YAN, Q., XU, Y. (2025): RLGS: Reinforcement learning-based adaptive hyperparameter tuning for Gaussian Splatting [online preprint]. arXiv:2508.04078. doi:10.48550/arXiv.2508.04078.

LIU, Y.-C., HÖLLEIN, L., NIEßNER, M., DAI, A. (2025): QuickSplat: Fast 3D surface reconstruction via learned Gaussian initialization. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 27851–27861. doi:10.1109/ICCV51701.2025.02585.

MCKAY, M. D., BECKMAN, R. J., CONOVER, W. J. (1979): A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics, 21(2), pp. 239–245. doi:10.1080/00401706.1979.10489755.

MILDENHALL, B., SRINIVASAN, P. P., TANCIK, M., BARRON, J. T., RAMAMOORTHI, R., NG, R. (2020): NeRF: Representing scenes as neural radiance fields for view synthesis. In: European Conference on Computer Vision. Lecture Notes in Computer Science. Cham: Springer, pp. 405–421. doi:10.1007/978-3-030-58452-8_24.

MÜLLER, T., EVANS, A., SCHIED, C., KELLER, A. (2022): Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics, 41(4), Article 102, pp. 1–15. doi:10.1145/3528223.3530127.

NEX, F., REMONDINO, F. (2014): UAV for 3D mapping applications: A review. Applied Geomatics, 6(1), pp. 1–15. doi:10.1007/s12518-013-0120-x.

PATEUX, S., GENDRIN, M., MORIN, L., LADUNE, T., JIANG, X. (2025): BOGausS: Better optimized Gaussian Splatting. In: Proceedings of the 33rd European Signal Processing Conference, pp. 765–769. doi:10.23919/EUSIPCO63237.2025.11226626.

PIX4D DOCUMENTATION [online] (n.d.): Example projects – real photogrammetry data: Pix4D test – Office building (DJI); Pix4D test – Fisheye Industrial building (Parrot Bebop 2); Pix4D test – Forensic Dataset; Pix4D test – Eagle statue (handheld camera); Pix4D test – Urban Area. Lausanne, Switzerland: Pix4D. [cit. 2026-06-18]. Available online: https://support.pix4d.com/hc/en-us/articles/360000235126.

RASMUSSEN, C. E., WILLIAMS, C. K. I. (2006): Gaussian processes for machine learning. Cambridge, MA: MIT Press. ISBN 978-0-262-18253-9.

SCHÖNBERGER, J. L., FRAHM, J.-M. (2016): Structure-from-motion revisited. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4104–4113. doi:10.1109/CVPR.2016.445.

SCHREIBER-GREGORY, D. N. [online] (2018): Ridge regression and multicollinearity: An in-depth review. In: Proceedings of SAS Global Forum 2018. [cit. 2026-06-18]. Available online: https://support.sas.com/resources/papers/proceedings18/1408-2018.pdf.

SHENDE, R., MUDUR, S., POPA, T., MENDHURWAR, K. (2024): Heritage3DMTL: An open multi-modal UAV dataset of Montreal's diverse heritage buildings for research [dataset]. Zenodo repository. doi:10.5281/zenodo.13732644.

SMOLA, A. J., SCHÖLKOPF, B. (2004): A tutorial on support vector regression. Statistics and Computing, 14(3), pp. 199–222. doi:10.1023/B:STCO.0000035301.49549.88.

SNOEK, J., LAROCHELLE, H., ADAMS, R. P. (2012): Practical Bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems 25, pp. 2951–2959.

SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., SUTSKEVER, I., SALAKHUTDINOV, R. (2014): Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(56), pp. 1929–1958.

TEPPATI LOSÈ, L., CHIABRANDO, F., GIULIO TONOLO, F., LINGUA, A. (2021): UAV photogrammetry and VHR satellite imagery for emergency mapping: The October 2020 flood in Limone Piemonte, Italy. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLIII-B3-2021, pp. 727–734. doi:10.5194/isprs-archives-XLIII-B3-2021-727-2021.

TIBSHIRANI, R. (1996): Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B, 58(1), pp. 267–288. doi:10.1111/j.2517-6161.1996.tb02080.x.

VÉLEZ, S., VACAS, R., MARTÍN, H., RUANO-ROSA, D., ÁLVAREZ, S. (2022): UAV RGB imagery dataset captured at nadir and oblique angles over pistachio trees in Spain, including images, GCPs, 3D point cloud and orthomosaic [dataset]. Zenodo repository. doi:10.5281/zenodo.7271542.

WANG, Z., BOVIK, A. C., SHEIKH, H. R., SIMONCELLI, E. P. (2004): Image quality assessment: From error visibility to structural similarity. IEEE Transactions on Image Processing, 13(4), pp. 600–612. doi:10.1109/TIP.2003.819861.

YE, V., LI, R., KERR, J., TURKULAINEN, M., YI, B., PAN, Z., SEISKARI, O., YE, J., HU, J., TANCIK, M., KANAZAWA, A. (2025): gsplat: An open-source library for Gaussian Splatting. Journal of Machine Learning Research, 26(34), pp. 1–17.

YU, Y., VERBREE, E., VAN OOSTEROM, P., POTTGIESSER, U. (2025): 3D Gaussian Splatting for modern architectural heritage: Integrating UAV-based data acquisition and advanced photorealistic 3D techniques. AGILE GIScience Series, 6, 51. doi:10.5194/agile-giss-6-51-2025.

YU, Z., CHEN, A., HUANG, B., SATTLER, T., GEIGER, A. (2024): Mip-Splatting: Alias-free 3D Gaussian Splatting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 19447–19456. doi:10.1109/CVPR52733.2024.01839.

ZHANG, R., ISOLA, P., EFROS, A. A., SHECHTMAN, E., WANG, O. (2018): The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 586–595. doi:10.1109/CVPR.2018.00068.

ZOU, H., HASTIE, T. (2005): Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B, 67(2), pp. 301–320. doi:10.1111/j.1467-9868.2005.00503.x.

APPENDIXES

Appendix 1 Datasets

Note: The platform identifier is the identifier used in the platform and reporting.

ID	Project name	Platform identifier	Dataset group	Study partition	Images	Country	Original resolution	Data source
1	Montreal City Hall	1-Montreal-City-Hall	Heritage3DMTL	Model-development	247	Canada	4032 × 3024	Heritage3DMTL dataset
2	Holy Ghost Ukrainian Church	10_Holy_Ghost_Ukrainian_Church	Heritage3DMTL	Model-development	101	Canada	1280 × 720	Heritage3DMTL dataset
3	St. George Church	11-St-George-Church	Heritage3DMTL	Model-development	66	Canada	4032 × 3024	Heritage3DMTL dataset
4	Murugan Temple	13-Murugan-Temple	Heritage3DMTL	Model-development	202	Canada	4032 × 3024	Heritage3DMTL dataset
5	Hindu Mandir	14-Hindu-Mandir	Heritage3DMTL	Model-development	246	Canada	4032 × 3024	Heritage3DMTL dataset
6	Congregation Shaar Hashomayim	15-Congregation-Shaar-Hashomayim	Heritage3DMTL	Model-development	117	Canada	4032 × 3024	Heritage3DMTL dataset
7	Mosque	16-Mosque	Heritage3DMTL	Model-development	174	Canada	4032 × 3024	Heritage3DMTL dataset
8	Shaughnessy Mansion	17-Shaughnessy-Mansion	Heritage3DMTL	Model-development	150	Canada	4032 × 3024	Heritage3DMTL dataset
9	Resurrection Chapel	2-Resurrection-Chapel	Heritage3DMTL	Model-development	272	Canada	4032 × 3024	Heritage3DMTL dataset
10	Maison Louis-Hippolyte La Fontaine	3-Maison_Louis-Hippolyte_Lafontaine	Heritage3DMTL	Model-development	107	Canada	4032 × 3024	Heritage3DMTL dataset
11	Gurdwara Lasalle	5-Gurdwara-Lasalle	Heritage3DMTL	Model-development	170	Canada	4032 × 3024	Heritage3DMTL dataset

12	Westmount City Hall	6-Westmount-City-Hall	Heritage3DMTL	Model-development	65	Canada	4000 × 2250	Heritage3DMTL dataset
13	Clock Tower	7-Clock-Tower	Heritage3DMTL	Model-development	219	Canada	4032 × 3024	Heritage3DMTL dataset
14	Saint Joseph’s Oratory	8-Saint-Joseph-Oratory	Heritage3DMTL	Model-development	243	Canada	4032 × 3024	Heritage3DMTL dataset
15	Loyola Chapel	9-Loyola-Chapel	Heritage3DMTL	Model-development	181	Canada	1280 × 720	Heritage3DMTL dataset
16	Bílovec	BILOVEC	—	Model-development	373	Czech Republic	5472 × 3648	Department of Geoinformatics, UPOL
17	Blume Drone Dataset	Blume_drone_data_capture_may2021	UAVID3D	Model-development	169	United States	5280 × 3956	UAVID3D dataset
18	Chapel of the Holy Trinity	Chapel_of_the_Holy_Trinity	—	Model-development	149	Czech Republic	5280 × 3956	National Heritage Institute
19	Château Dívčí Hrad 20 nadir and 60 degrees	Chatteau	Château Dívčí Hrad	Model-development	169	Czech Republic	4280 × 3956	Department of Geoinformatics, UPOL
20	Concord Buchanan Airport FAA Tower	Concord_Buchanan_Airport_FAA_Tower	Esri sample datasets	Model-development	200	United States	5472 × 3648	Esri sample drone datasets
21	KTM20 Oblique – Flight 003	DJI_202402171413_003_KTM20-oblique	KTM20 Oblique	Model-development	457	Nepal	5280 × 3956	Geovation Nepal
22	KTM20 Oblique – Flight 004, Group 1	DJI_202402171440_004_KTM20-oblique-split_group_1	KTM20 Oblique	Model-development	430	Nepal	5280 × 3956	Geovation Nepal
23	KTM20 Oblique – Flight 004, Group 2	DJI_202402171440_004_KTM20-oblique-split_group_2	KTM20 Oblique	Model-development	429	Nepal	5280 × 3956	Geovation Nepal
24	KTM20 Oblique – Flight 005	DJI_202402171501_005_KTM20-oblique	KTM20 Oblique	Model-development	469	Nepal	5280 × 3956	Geovation Nepal
25	KTM20 Oblique – Flight 008	DJI_202402171525_008_KTM20-oblique	KTM20 Oblique	Model-development	399	Nepal	5280 × 3956	Geovation Nepal

26	Red Rocks Amphitheatre – Group 1	Denver_Red_Rocks_Amphitheatre_Split_group_1	Red Rocks Amphitheatre	Model-development	137	United States	9504 × 6336	Esri sample drone datasets
27	Red Rocks Amphitheatre – Group 2	Denver_Red_Rocks_Amphitheatre_Split_group_2	Red Rocks Amphitheatre	Model-development	172	United States	9504 × 6336	Esri sample drone datasets
28	Gaurisankar	Gaurisankar	—	Model-development	84	Nepal	4000 × 3000	Geovation Nepal
29	Jindřichův Hradec St. James	Jindrichuv_Hradec_St_JakubUAS	—	Model-development	406	Czech Republic	5472 × 3078	National Heritage Institute
30	Trakai Island Castle – Group 1	Lithuania_Trakai_Island_Castle_Split_group_1	Trakai Island Castle	Model-development	187	Lithuania	8192 × 5460	Esri sample drone datasets
31	Trakai Island Castle – Group 2	Lithuania_Trakai_Island_Castle_Split_group_2	Trakai Island Castle	Model-development	173	Lithuania	8192 × 5460	Esri sample drone datasets
32	Prejmer	Mavic2DET_Prejmer	—	Model-development	162	Romania	4056 × 3040	Department of Geoinformatics, Paris Lodron University Salzburg
33	Naklo	NAKLO	—	Model-development	156	Czech Republic	4536 × 8034	Department of Geoinformatics, UPOL
34	Narayani Bridge	Narayani_Bridge	—	Model-development	891	Nepal	5280 × 3956	Geovation Nepal
35	Otaslavice	Otaslavice	—	Model-development	395	Czech Republic	960 × 544	Department of Geoinformatics, UPOL
36	Pistachio Trees	Pistachio-trees-by-velez	Vélez et al. dataset	Model-development	248	Spain	5472 × 3078	Vélez et al. UAV RGB imagery dataset
37	Pix4D Urban Area	Pix4Dmatic_example_100_images	Pix4D sample datasets	Model-development	100	Not specified	6000 × 4000	Pix4D example datasets
38	Pix4D Office Building	Pix4d_Demo_Office_Building	Pix4D sample datasets	Model-development	85	Not specified	4000 × 3000	Pix4D example datasets

39	Pix4D Eagle Statue	Pix4d_eagle	Pix4D sample datasets	Model-development	44	Switzerland	5184 × 3456	Pix4D example datasets
40	Telč St. James – 19 March	Telc_sv_Jakub_March_19	Telč St. James	Model-development	292	Czech Republic	5472 × 3078	National Heritage Institute
41	Angel Island State Park – Group 1	Tiburon_Angel_Islands_State_Park_Split_group_1	Angel Island State Park	Model-development	261	United States	9504 × 6336	Esri sample drone datasets
42	Angel Island State Park – Group 2	Tiburon_Angel_Islands_State_Park_Split_group_2	Angel Island State Park	Model-development	268	United States	9504 × 6336	Esri sample drone datasets
43	Angel Island State Park – Group 4	Tiburon_Angel_Islands_State_Park_Split_group_4	Angel Island State Park	Model-development	212	United States	9504 × 6336	Esri sample drone datasets
44	Huzová	huzova	—	Model-development	287	Czech Republic	4000 × 3000	Department of Geoinformatics, UPOL
45	Podolí	podoli	—	Model-development	69	Czech Republic	4000 × 3000	Department of Geoinformatics, UPOL
46	Správek	spranek	—	Model-development	643	Czech Republic	4000 × 3000	Department of Geoinformatics, UPOL
47	Maisonneuve Market	12-Maisonneuve Market	Heritage3DMTL	Final test	394	Not specified	4032 × 3024	Heritage3DMTL dataset
48	Thomas More Church	4-Thomas-More-Church	Heritage3DMTL	Final test	158	Not specified	4032 × 3024	Heritage3DMTL dataset
49	Brasov	Brasov	—	Final test	888	Romania	4000 × 3000	Department of Geoinformatics, Paris Lodron University Salzburg
50	Changunarayan Temple	Changu	—	Final test	1,048	Nepal	4056 × 3040	Geovation Nepal
51	Château Dívčí Hrad – Circular Flight	Chatteau_circle_60_and_45_degrees	Château Dívčí Hrad	Final test	44	Czech Republic	4280 × 3956	Department of Geoinformatics, UPOL

52	KTM20 Oblique – Flight 006	DJI_202402171501_006_KTM20-oblique	KTM20 Oblique	Final test	478	Nepal	5280 × 3956	Geovation Nepal
53	Knínice Church of the Exaltation of the Holy Cross	Kninice_Church_of_the_Exaltation_of_the_Holy_Cross	—	Final test	160	Czech Republic	5472 × 3078	National Heritage Institute
54	Pix4D Forensic	Pix4d_forensic	Pix4D sample datasets	Final test	69	Not specified	4000 × 3000	Pix4D example datasets
55	Telč St. James – 18 March	Telc_sv_Jakub_March_18	Telč St. James	Final test	212	Czech Republic	5472 × 3078	National Heritage Institute
56	Angel Island State Park – Group 3	Tiburon_Angel_Islands_State_Park_Split_group_3	Angel Island State Park	Final test	304	United States	9504 × 6336	Esri sample drone datasets
57	Morice	morice	—	Final test	246	Czech Republic	4000 × 3000	Department of Geoinformatics, UPOL
58	Unčovice	uncovice	—	Final test	956	Czech Republic	4000 × 3000	Department of Geoinformatics, UPOL

Appendix 2 Coefficients of the final Ridge Regression scoring model

Index	Model term	Coefficient	Index		Coefficient
0	Intercept	−0.010702	24	Terrain-roughness proxy × geometry multiplier	−0.008870
1	GSD median	+0.004997	25	Vegetation-complexity score × geometry multiplier	+0.065471
2	Overlap proxy	+0.001109	26	Vegetation-cover percentage × geometry multiplier	−0.068197
3	Coverage spread	−0.013102	27	GSD median × appearance multiplier	−0.002254
4	Camera-angle bucket	0.000000	28	Overlap proxy × appearance multiplier	+0.001013
5	Heading consistency	+0.000914	29	Coverage spread × appearance multiplier	−0.008118
6	Texture density	−0.028993	30	Camera-angle bucket × appearance multiplier	0.000000
7	Blur-motion risk	−0.020295	31	Heading consistency × appearance multiplier	+0.000143
8	Terrain-roughness proxy	+0.000986	32	Texture density × appearance multiplier	+0.018567
9	Vegetation-complexity score	+0.003012	33	Blur-motion risk × appearance multiplier	+0.013072
10	Vegetation-cover percentage	−0.001036	34	Terrain-roughness proxy × appearance multiplier	−0.010986
11	Geometry multiplier	+0.042715	35	Vegetation-complexity score × appearance multiplier	−0.014206
12	Appearance multiplier	+0.020156	36	Vegetation-cover percentage × appearance multiplier	+0.012585
13	Densification multiplier	−0.018694	37	GSD median × densification multiplier	+0.011254
14	Squared geometry multiplier	−0.037237	38	Overlap proxy × densification multiplier	−0.003390
15	Squared appearance multiplier	−0.036959	39	Coverage spread × densification multiplier	+0.014858
16	Squared densification multiplier	+0.344883	40	Camera-angle bucket × densification multiplier	0.000000
17	GSD median × geometry multiplier	+0.003868	41	Heading consistency × densification multiplier	−0.003553
18	Overlap proxy × geometry multiplier	−0.003827	42	Texture density × densification multiplier	−0.037253

19	Coverage spread × geometry multiplier	−0.008084	43	Blur-motion risk × densification multiplier	−0.025070
20	Camera-angle bucket × geometry multiplier	0.000000	44	Terrain-roughness proxy × densification multiplier	+0.016260
21	Heading consistency × geometry multiplier	−0.001530	45	Vegetation-complexity score × densification multiplier	−0.013847
22	Texture density × geometry multiplier	+0.013765	46	Vegetation-cover percentage × densification multiplier	+0.016739
23	Blur-motion risk × geometry multiplier	+0.017563			

Note: The camera-angle bucket and its interaction terms have coefficients of zero because this descriptor showed no variation among the model-development projects. All projects were assigned to the same camera-angle category.

Appendix 3 Model Results

Appendix 3A Ridge Regression quality results and selected multiplier combinations

Note: The Ridge-selected run for KTM20 Oblique – Flight 006 reached the Gaussian hard cap at training step 3,900 and did not produce complete final PSNR, SSIM, and LPIPS values.

No.	Project	Status	Geometry	Appearance	Densification	Baseline PSNR	Ridge PSNR	ΔPSNR (dB)	Baseline SSIM	Ridge SSIM	ΔSSIM	Baseline LPIPS	Ridge LPIPS	LPIPS improvement
1	Maisonneuve Market	Completed	1.475	1.057	1.430	17.098	16.975	−0.123	0.653	0.656	+0.002	0.474	0.467	+0.007
2	Thomas More Church	Completed	1.475	1.181	1.430	21.867	21.504	−0.364	0.780	0.783	+0.003	0.302	0.300	+0.002
3	Brasov	Completed	2.057	1.057	1.430	23.362	23.368	+0.006	0.802	0.816	+0.013	0.193	0.167	+0.025
4	Changunarayan Temple	Completed	1.648	1.320	0.700	19.980	20.078	+0.097	0.622	0.644	+0.022	0.518	0.467	+0.052
5	Château Dívčí Hrad – Circular Flight	Completed	2.057	1.475	0.700	23.375	23.805	+0.429	0.568	0.654	+0.086	0.449	0.252	+0.197
6	KTM20 Oblique – Flight 006	Hard cap	2.569	1.181	0.700	22.672	—	—	0.745	—	—	0.309	—	—
7	Knínice Church of the Exaltation of the Holy Cross	Completed	1.841	1.181	0.700	20.943	21.209	+0.266	0.683	0.707	+0.024	0.454	0.396	+0.057
8	Pix4D Forensic	Completed	1.648	1.648	0.700	23.920	25.036	+1.116	0.521	0.672	+0.151	0.649	0.414	+0.235
9	Telč St. James – 18 March	Completed	1.648	1.320	0.700	21.139	21.101	−0.038	0.708	0.739	+0.031	0.432	0.361	+0.071
10	Angel Island State Park – Group 3	Completed	1.648	1.181	0.700	20.191	20.332	+0.141	0.638	0.674	+0.036	0.514	0.441	+0.074
11	Morice	Completed	1.648	1.320	0.700	28.495	29.475	+0.980	0.800	0.858	+0.058	0.274	0.168	+0.106
12	Unčovice	Completed	1.648	1.648	0.700	22.647	22.925	+0.277	0.659	0.702	+0.043	0.513	0.419	+0.093

Appendix 3B Ridge Regression Gaussian-count and runtime results

No.	Project	Baseline Gaussians	Ridge Gaussians	Δ Gaussians	Baseline runtime (s)	Ridge runtime (s)	Δ Runtime (s)
1	Maisonneuve Market	317,276	237,468	−79,808	429.9	428.8	−1.1
2	Thomas More Church	667,244	508,632	−158,612	380.7	374.1	−6.7
3	Brasov	3,107,384	2,665,212	−442,172	999.7	1,058.9	+59.2
4	Changunarayan Temple	892,774	1,787,759	+894,985	762.4	869.9	+107.5
5	Château Dívčí Hrad – Circular Flight	414,895	2,297,891	+1,882,996	302.4	615.8	+313.4
6	KTM20 Oblique – Flight 006	3,205,138	6,227,049*	+3,021,911	922.2	1,494.8*	+572.6
7	Knínice Church of the Exaltation of the Holy Cross	295,517	718,806	+423,289	277.8	356.7	+78.9
8	Pix4D Forensic	377,794	2,369,708	+1,991,914	309.8	585.1	+275.4
9	Telč St. James – 18 March	895,889	2,437,644	+1,541,755	367.8	629.0	+261.2
10	Angel Island State Park – Group 3	1,081,005	2,512,788	+1,431,783	472.6	734.9	+262.3
11	Morice	918,084	2,891,199	+1,973,115	476.4	849.0	+372.6
12	Unčovice	424,305	1,450,299	+1,025,994	425.7	542.2	+116.5

* The Ridge-selected run for Project 6 was stopped after exceeding the Gaussian hard cap. Its Gaussian count and runtime represent the values recorded when the run was stopped, not a normally completed result.

Appendix 3C MLP quality results and selected multiplier combinations

No.	Project	Status	Geometry	Appearance	Densification	Baseline PSNR	MLP PSNR	ΔPSNR (dB)	Baseline SSIM	MLP SSIM	ΔSSIM	Baseline LPIPS	MLP LPIPS	LPIPS improvement
1	Maisonneuve Market	Completed	1.320	1.320	0.700	17.098	16.420	−0.679	0.653	0.649	−0.004	0.474	0.479	−0.005
2	Thomas More Church	Completed	1.181	1.475	0.700	21.867	21.801	−0.067	0.780	0.806	+0.026	0.302	0.246	+0.056
3	Brasov	Hard cap	1.841	1.181	0.700	23.362	—	—	0.802	—	—	0.193	—	—
4	Changunarayan Temple	Completed	1.475	1.475	0.700	19.980	20.104	+0.123	0.622	0.653	+0.031	0.518	0.455	+0.064
5	Château Dívčí Hrad – Circular Flight	Completed	2.569	1.648	0.700	23.375	23.727	+0.352	0.568	0.661	+0.093	0.449	0.234	+0.216
6	KTM20 Oblique – Flight 006	Hard cap	2.299	1.057	0.700	22.672	—	—	0.745	—	—	0.309	—	—
7	Knínice Church of the Exaltation of the Holy Cross	Completed	1.648	1.648	0.700	20.943	21.276	+0.333	0.683	0.714	+0.032	0.454	0.383	+0.071
8	Pix4D Forensic	Completed	2.870	1.475	0.700	23.920	25.115	+1.195	0.521	0.678	+0.157	0.649	0.391	+0.258
9	Telč St. James – 18 March	Completed	1.320	1.475	0.700	21.139	21.609	+0.470	0.708	0.740	+0.032	0.432	0.363	+0.069
10	Angel Island State Park – Group 3	Completed	2.299	1.181	0.700	20.191	20.080	−0.111	0.638	0.670	+0.032	0.514	0.435	+0.079
11	Morice	Completed	2.057	1.475	0.700	28.495	29.565	+1.071	0.800	0.867	+0.067	0.274	0.150	+0.124
12	Unčovice	Completed	1.841	1.648	0.700	22.647	22.574	−0.074	0.659	0.696	+0.037	0.513	0.429	+0.084

Note: The MLP-selected runs for Brasov and KTM20 Oblique – Flight 006 reached the Gaussian hard cap and did not produce complete final PSNR, SSIM, and LPIPS values.

Appendix 3D MLP Gaussian-count and runtime results

No.	Project	Baseline Gaussians	MLP Gaussians	Δ Gaussians	Baseline runtime (s)	MLP runtime (s)	Δ Runtime (s)
1	Maisonneuve Market	317,276	606,702	+289,426	429.9	504.8	+74.9
2	Thomas More Church	667,244	1,921,140	+1,253,896	380.7	586.8	+206.1
3	Brasov	3,107,384	6,014,753*	+2,907,369	999.7	1,657.3*	+657.6
4	Changunarayan Temple	892,774	2,063,088	+1,170,314	762.4	927.7	+165.3
5	Château Dívčí Hrad – Circular Flight	414,895	2,836,880	+2,421,985	302.4	720.0	+417.6
6	KTM20 Oblique – Flight 006	3,205,138	6,018,142*	+2,813,004	922.2	1,266.7*	+344.5
7	Knínice Church of the Exaltation of the Holy Cross	295,517	1,020,729	+725,212	277.8	385.2	+107.4
8	Pix4D Forensic	377,794	2,016,809	+1,639,015	309.8	527.1	+217.3
9	Telč St. James – 18 March	895,889	2,582,481	+1,686,592	367.8	645.4	+277.6
10	Angel Island State Park – Group 3	1,081,005	2,423,993	+1,342,988	472.6	682.4	+209.8
11	Morice	918,084	3,478,344	+2,560,260	476.4	864.9	+388.6
12	Unčovice	424,305	1,438,100	+1,013,795	425.7	581.5	+155.8

* The MLP-selected runs for Projects 3 and 6 were stopped after exceeding the Gaussian hard cap. Their Gaussian counts and runtimes represent the values recorded when the runs were stopped.

Appendix 3E Checkpoint Comparison OF Baseline And Ridge Model-Selected Runs

No.	Project	Base PSNR 5k	Ridge PSNR 5k	ΔPSNR 5k	Base PSNR 7k	Ridge PSNR 7k	ΔPSNR 7k	Base SSIM 5k	Ridge SSIM 5k	ΔSSIM 5k	Base SSIM 7k	Ridge SSIM 7k	ΔSSIM 7k	Base LPIPS 5k	Ridge LPIPS 5k	LPIPS impr. 5k	Base LPIPS 7k	Ridge LPIPS 7k	LPIPS impr. 7k
1	Maisonneuve Market	16.900	16.983	+0.083	17.225	17.500	+0.275	0.648	0.653	+0.005	0.656	0.664	+0.007	0.488	0.476	+0.012	0.466	0.450	+0.016
2	Thomas More Church	21.401	21.941	+0.540	22.203	22.191	−0.012	0.780	0.792	+0.012	0.796	0.800	+0.004	0.295	0.285	+0.010	0.269	0.267	+0.002
3	Brasov	23.465	23.419	−0.046	24.209	24.074	−0.135	0.812	0.817	+0.005	0.829	0.835	+0.007	0.181	0.166	+0.014	0.158	0.141	+0.016
4	Changunarayan Temple	20.063	19.974	−0.089	20.425	20.405	−0.021	0.623	0.633	+0.010	0.641	0.660	+0.019	0.516	0.488	+0.029	0.483	0.439	+0.044
5	Château Dívčí Hrad – Circular Flight	23.399	23.898	+0.499	23.385	23.730	+0.345	0.585	0.666	+0.081	0.590	0.662	+0.073	0.420	0.232	+0.188	0.401	0.222	+0.178
6	KTM20 Oblique – Flight 006	22.746	21.253	−1.494	23.475	22.662	−0.813	0.752	0.706	−0.046	0.785	0.779	−0.007	0.306	0.330	−0.024	0.254	0.233	+0.021
7	Knínice Church	21.227	21.547	+0.320	21.815	22.039	+0.224	0.688	0.711	+0.024	0.696	0.722	+0.026	0.447	0.391	+0.056	0.428	0.367	+0.062
8	Pix4D Forensic	24.260	24.814	+0.554	24.417	24.993	+0.576	0.543	0.673	+0.131	0.556	0.691	+0.135	0.623	0.405	+0.219	0.596	0.363	+0.234
9	Telč – St. James Church	21.058	21.304	+0.245	21.489	21.669	+0.180	0.715	0.745	+0.030	0.728	0.761	+0.033	0.421	0.360	+0.061	0.394	0.325	+0.069
10	Angel Island State Park	20.217	20.082	−0.136	20.508	20.402	−0.106	0.644	0.665	+0.021	0.662	0.693	+0.031	0.506	0.457	+0.049	0.470	0.401	+0.069
11	Morice	28.537	29.513	+0.977	28.992	30.312	+1.320	0.813	0.865	+0.052	0.823	0.876	+0.053	0.252	0.156	+0.096	0.234	0.137	+0.097
12	Unčovice	22.643	22.163	−0.480	23.248	22.486	−0.763	0.663	0.688	+0.025	0.675	0.703	+0.028	0.506	0.442	+0.064	0.481	0.409	+0.072

Note: “Base” refers to the baseline run, while “Ridge” refers to the Ridge model-selected run. PSNR and SSIM differences were calculated as the Ridge model-selected value minus the corresponding baseline value. LPIPS improvement was calculated as the baseline value minus the Ridge model-selected value because lower LPIPS indicates better quality. Positive values therefore indicate improved performance by the Ridge model-selected run. The 5,000- and 7,000-iteration values were obtained from the same continuous training runs within the separate 7,000-iteration pipeline. These runs were not continuations of the main 5,000-iteration evaluation. All other processing settings remained unchanged, and densification ended at 4,000 iterations in both cases.

Appendix 3F Supporting Models Trained With Hard-Cap Data

Supporting model	Completed normally	Hard-cap runs	PSNR improved	SSIM improved	LPIPS improved	All three improved	Mean PSNR difference	Mean SSIM difference	Mean LPIPS improvement
Ridge trained with hard-cap data	12/12	0	4/12	6/12	7/12	4/12	-0.086 dB	+0.0032	+0.0074
MLP trained with hard-cap data	12/12	0	6/12	8/12	10/12	6/12	-0.145 dB	+0.0158	+0.0439

Appendix 3G Repeatability Check Using the Same Multiplier Combinations

Run	Completed normally	Hard-cap runs	PSNR improved	SSIM improved	LPIPS improved	All three improved	Mean PSNR diff	Median PSNR diff	Mean SSIM diff	Median SSIM diff	Mean LPIPS improvement	Median LPIPS improvement
First 3DGS run with selected multipliers	11/12	1	8/11	11/11	11/11	8/11	+0.2536 dB	+0.2662 dB	+0.0428	+0.0361	+0.0835	+0.0737
Second 3DGS run with same selected multipliers	11/12	1	8/11	10/11	10/11	8/11	+0.2120 dB	+0.1948 dB	+0.0417	+0.0363	+0.0820	+0.0752

Attachments

The attachments includes the thesis text and poster, a backup of the thesis website, a backup of the interactive results platform, the Bimba3D-re source-code backup, reported research outputs, and selected visualization files.

SN	Folder Name	Contents
1	1. Thesis Report and Poster	Final thesis report PDF and final thesis poster PDF.
2	2. Website	Backup of the thesis website, including HTML, CSS, JavaScript, figures, logos, downloads, and assets required to run the static website.
3	3. Research Outputs	Reported result tables, model artifacts, checkpoint metrics, selected previews, and traceability files from the Ridge Regression and MLP experiments. See the README and file_manifest.csv inside this folder for detailed folder structure.
4	4. Splats and rad 3d files	Selected original .splat checkpoint files and converted browser-ready RAD visualization files. The rag subfolder name is kept as the local conversion-output folder name, but its contents are RAD-format visualization files.
5	5. Bimba3D-re Source Code	Source-code backup of the Bimba3D-re platform developed for the thesis workflow.
6	6. Results Platform Source Code	Source-code backup of the interactive results platform used to present selected public cases, charts, comparisons, and 3D visualization links.

Data-Sharing Scope

The attachments do not include every file produced during processing. Only report-facing outputs, selected checkpoint data, selected visualization files, source-code backups, and traceability files are included. Complete raw image datasets and all intermediate reconstruction artifacts are not redistributed because some datasets are private or institutionally provided, and full processing outputs are very large. Public source datasets are referenced in the thesis report where applicable.