Towards An Intelligent Data Acquisition System for UAV-based 3D Reconstruction

by

Sara Hatami Gaznai

B.Sc., University of Tehran, 2021

A Thesis Submitted In Partial Fulfillment of

the Requirements for The Degree of

MASTER OF APPLIED SCIENCE

in

The Department of Mechanical Engineering

University of Victoria

Towards An Intelligent Data Acquisition System for UAV-based 3D Reconstruction

by

Sara Hatami Gaznai

B.Sc., University of Tehran, 2021

Supervisory Committee

Dr. Homayoun Najjaran, Supervisor

(Department of Mechanical Engineering)

Dr. Hong-Chuan Yang, Supervisory Committee Member

(Department of Electrical and Computer Engineering)

ABSTRACT

Uncrewed Aerial Vehicles (UAVs) have become a pivotal platform for acquiring data in various applications, particularly for inspection, monitoring, and modeling purposes. However, the limited flight time and energy consumption of UAVs have necessitated the development of intelligent data acquisition systems for these platforms. In cases where there is no geometric proxy of the target and its location is unknown, it becomes crucial to establish a model that enables the detection of the target through visual data. Subsequently, the UAV incrementally acquires data as it navigates around the target, utilizing onboard sensors to complete its interpretation of the target or the scene. To accomplish this, the UAV adheres to a strategy known as view planning, which plays a critical role in three dimensional (3D) reconstruction of infrastructure using UAV-based imaging and significantly influences the quality of the reconstruction results. The selected views to be captured must essentially reveal the most unknown information about the target to ensure efficiency as well as utility. In this work, we propose three essential components of an intelligent data acquisition system: i) identification of the optimal views to prioritize, ii) multi-task sensor fusion for depth completion and object detection, and iii) reinforcement learning of appearance-based next-best-view (NBV) planning. The main focus of this study is to establish a relationship between the visual features observed in 2D images and the corresponding 3D model of the target, aiming to avoid the computational cost of handling 3D data.

Contents

List of Tables

List of Figures

LAY SUMMARY

This thesis explores the development of a 3D perception system for aerial data acquisition, with a specific focus on 3D reconstruction. The main objective is to capture 2D images of infrastructure such as bridges and heritage buildings using drones and utilize these images to create and present 3D models. The limitations of energy and computation resources onboard the drone restrict the image acquisition process, which can impact the quality of the resulting 3D model if the resources are not efficiently utilized to capture valuable data. This research aims to optimize the data acquisition process in several aspects. First, the views that are of utmost utility for the reconstruction process are identified. Identifying and selecting these views reduces the number of captured views while maintaining the high quality of the generated 3D model. Then, a multi-task model is introduced for efficient use of sensor data intended to produce higher quality inputs for 3D reconstruction models. These two components ensure that the captured views contain valuable information and are of high resolution, while minimizing the cost associated with capturing such views. Finally, a view planning model is developed to sequentially identify and capture the aforementioned high-utility views in an online manner. Overall, the aim is to streamline the 3D reconstruction process and reduce the number of views required to achieve a high-quality reconstruction.

PREFACE

This thesis is the result of a collaboration between the Advanced Control and Intelligent Systems (ACIS) Laboratory at the University of Victoria and the National Research Council (NRC) of Canada through the Artificial Intelligence for Logistics (AI4L) program under the grant agreement DHGA AI4L-129-2 (CDB #6835).

Chapter 2 includes concepts and material that have been published on the proceedings of *IEEE SMC 2022*. This material exist in all the sections of this chapter excluding the material related to the Bag-of-Views method and dataset refinement experiments. Chapter 2 and Chapter 4 include material that has been published on *arXiv* as a paper pre-print which has been submitted to a journal for possible publication. This material includes the Bag-of-Views method, the proposed view planning method and the corresponding experiments and discussions.

The material in Chapter 3 on the proposed multi-task approach to aerial depth completion was accepted to *IEEE SMC 2023* for publication. A pre-print version of this paper is also available on *arXiv*.

In all the abovementioned works, I, Sara Hatami Gazani, was responsible for concept development, research, implementation, and experiments for all the material included in this thesis. Matthew Tucsok was involved in developing the simulation environment of the work in Chapters 2 and 4 and contributed to the programming stage and manuscript formation. Fardad Dadboud contributed to programming for the implementation of the multi-task network in Chapter 3 and writing and formatting the manuscript. Iraj Mantegh was involved in the concept formation and editing the manuscript. Homayoun Najjaran, the supervisory author, was involved throughout all phases of the project in concept formation, supervision, and manuscript revision.

ABBREVIATIONS

Abbreviation	Full Meaning
2D, 3D	two-dimensional, three-dimensional
AC	actor-critic
BoV	bag-of-views
BoVW	bag-of-visual-words
BoW	bag-of-words
BRIEF	binary robust independent elementary features
CNN	convolutional neural network
DQN	deep Q-network
FAST	features from accelerated segment test
FPN	feature pyramid network
GCP	ground control points
GPS	global positioning system
HOG	histogram of oriented gradients
ICP	iterative closest point
IoU	intersection-over-union
IMU	inertial measurement unit
LiDAR	light detecting and ranging
LSTM	long short-term memory
MSE	mean squared error
MSBE	mean squared Bellman error
MC	Monte Carlo
MDP	Markov decision process
MD	Markov Process

MRP	Markov reward process
MTL	multi-task learning
NBV	next-best-view
NMS	non-maximum suppression
ORB	oriented FAST and rotated BRIEF
PrGAN	projective generative adversarial network
RCNN	region-based convolutional neural network
ReLU	rectified linear unit
RGB	red-green-blue color model
RMSE	root mean square error
ROI	region-of-interest
RPN	region proposal network
SAE	sparse autoencoder
SAC	soft actor-critic
SDF	signed distance field
SfM	Structure from Motion
SIFT	scale-invariant feature transform
SURF	speeded-up robust features
TD	temporal difference
TSDF	truncated signed distance function
UAV	uncrewed aerial vehicle
VAE	variational autoencoder

NOMENCLATURE

Symbol	Definition	Equation
K	camera intrinsic matrix	2.1
c_x, c_y	optical center offset x and y axes	2.1
s	axis skew	2.1
R	rotation matrix	2.2
T	translation matrix	2.2
f_x, f_y	focal length in x and y axes	2.2
(u, v)	pixel position in image coordinate frame	2.2
l_1	reconstruction loss	2.3
y_i	ground truth occupancy value for the	2.3
N	number of samples/batches in the dataset	2.3
σ_i	variance of the probability distribution for the	2.4
μ	mean of the probability distribution for the	2.4
p_i	predicted occupancy probability for the	2.4
l_2	kullback-Leibler Divergence Loss	2.4
i	index representing a particular view	2.5
j	index for iterating over feature descriptors	2.5
m	number of extracted features	2.5
χ_i	i_{th} view in the set	2.5

ν_{id}	vocabulary assigned to a specific view range with identifier id	2.6
p_i	output of the network representing the probability of occupancy	2.7
y_i	ground truth value of occupancy	2.7
θ	threshold used to binarize occupancy probabilities	2.7
$\mathbf{I}(\cdot)$	indicator function	2.7
$d_{CD}(R, P)$	Chamfer discrepancy between points in R and points in P	2.8
$\|R\|$	number of points in the reconstructed point cloud R	2.8
r	a point in the reconstructed point cloud	2.8
p	a point in the reference point cloud	2.8
$d_{HD}(R, P)$	Hausdorff distance between points in R and points in P	2.8
$d(R, P)$	Euclidean distance between points in R and points in P	2.9
sup	supremum or least upper bound operation	2.9
inf	infimum or greatest lower bound operation	2.9
$l_{\text{consistency}}$	loss for depth consistency	3.1
n	number of pixels	3.1
Y_i	ground truth depth value of the i^{th} pixel	3.1
$\hat{Y}_i$	predicted depth value of the i^{th} pixel	3.1
$l_{\text{smoothness}}$	loss for depth smoothness	3.2
$\partial_x^2 \hat{Y}_i$	second order derivative of predicted depth in the x-axis	3.2
$\partial_y^2 \hat{Y}_i$	second order derivative of predicted depth in the y-axis	3.2
λ	weighting factor for combining proposal and final detection losses	3.3
N_{cls}	normalization factor for classification loss	3.4
N_{reg}	normalization factor for regression loss	3.5
t_{ijk}	predicted box offset for the k^{th} parameter of the i^{th} proposal of the j^{th} anchor	3.5

t^*_{ijk}	ground truth box offset for the k^{th} parameter of the i^{th} proposal of the j^{th} anchor	3.5
$\text{smooth}_{L1}(.)$	smooth L1 loss	3.7
$p(S_{t+1}\|S_t)$	transition probability from state S_t to state S_{t+1} in a Markov Process (MP)	4.1
S_t	state at time t	4.1
R_{t+1}	reward received at time $t+1$	4.2
s	state value	4.2
a	action value	4.2
A_t	action at time t	4.3
$r(s', s, a)$	expected reward when transitioning to state s' from state s after taking action a	4.3
γ	discount factor for future rewards	4.4
G_t	discounted expected return	4.4
v	state value function	4.5
$V_\pi(s)$	expected cumulative discounted reward in state s under policy π	4.5
$\mathbb{E}_\pi$	expectation under policy π	4.5
q	action value function	4.6
$q_\pi(s, a)$	expected cumulative discounted reward starting from π state s taking action a, and following policy	4.6
$L_i(\theta_i)$	loss function at iteration i	4.11

U	experience replay	4.11
α	entropy regularization coefficient	4.13
π^*	optimal policy	4.15
$H(\pi(\cdot\|s_t))$	entropy of policy	4.15
$a \sim \pi$	action sampled from policy	4.16
$s' \sim P$	next state sampled from transition distribution	4.17
$\tilde{a}'$	sampled action	4.18
$\tilde{a}' \sim \pi(\cdot\|s')$	sampled action from policy	4.20
$\phi_{\text{targ},j}$	rarget Q-function parameters	4.20
$\tilde{a}_\theta(s, \xi)$	reparameterized action	4.21
$\xi \sim \mathcal{N}(0, I)$	sampled noise	4.21
$\mu_\theta(s)$	mean of action distribution	4.21
$\sigma_\theta(s)$	logarithm of standard deviation of action distribution	4.21
$\log \pi_\theta(\tilde{a}_\theta(s, \xi)\|s)$	log-probability of action under policy	4.22

ACKNOWLEDGEMENTS

I am sincerely grateful to Dr. Homayoun Najjaran from the Department of Mechanical Engineering at the University of Victoria for his unwavering support and encouragement throughout my Master's program. His trust in my research work fueled my motivation, fostering my curiosity and creativity.

I would also like to express my heartfelt appreciation to Matthew Tucsok from the Okanagan School of Engineering at the University of British Columbia. Our collaboration on this project was an unbelievably amazing teamwork experience.

I extend my gratitude to Dr. Iraj Mantegh of the National Research Council (NRC) Canada for his invaluable support and advice.

Lastly, I am indebted to my mother, who overcame the challenges of long distance, enabling me to pursue this Master's program.

DEDICATION

To my beloved brother, who made countless sacrifices to ensure that his not so little sister's dreams blossomed into reality.

Chapter 1

Introduction

Uncrewed Aerial Vehicles (UAVs) have emerged as a valuable tool in various fields, revolutionizing applications such as aerial photography, monitoring, and mapping. These platforms can carry the essential sensors to regions that are not accessible by ground vehicles and acquire valuable information for the specific application they are being used for. Among such applications, UAV-based 3D reconstruction of infrastructure holds immense potential. This application enables the creation of high-quality three-dimensional (3D) models of target infrastructure whose maintenance planning is highly dependant on awareness of their conditions and the change of these conditions in time. UAV-based data acquisition has also significantly influenced other domains such as urban planning, disaster response, and virtual reality, where high-fidelity 3D representations of environments are crucial.

There are several methods for maneuvering a drone, each serving different purposes and offering varying degrees of control. Some of these methods include but are not limited to manual control, intelligent flight systems, or the use of a mission planning software. In manual control, real-time control of the UAV is involved and is often used in situations where precise maneuvering is required. In this method, piloting the drone is done by using a controller or remote to control the drone's movements, including altitude, yaw (rotation), pitch (forward or backward movement), and roll

(sideways movement) [3]. A disadvantage of this mode for data acquisition applied in 3D reconstruction is that there might be parts of the target are that not visible to the drone pilot and even if they are, sequentially identifying them and maneuvering the drone to those regions is not time efficient due to the accumulated response time of the pilot which poses a challenge to the onboard energy resources and flight time limitations. With the Waypoint Navigation mode, the drone follows a designated path composed of specific Global Positioning System (GPS) coordinates [4]. Although this mode offers consistency and precision in attending the pre-defined waypoints, it lacks flexibility in reacting to dynamic environments and decision making abilities towards satisfying the specific requirements of the task it is used for. In this regard, using a mission planning software allows more flexibility for the drone mission and the user can specify actions to be done at each location and set flight parameters [5]. The suitability of the planned mission for the specific application in this case requires previous knowledge of the environment that the drone will be investigating.

With the recent advances in artificial intelligence, intelligence flight systems have gained an increasing attention across various industries and sectors. These systems leverage advanced algorithms in computer vision, sensor fusion, and path planning domains to operate autonomously in dynamic, GPS-denied or unknown environments. This thesis focuses on the developement of an intelligent data acquisition system targeting the application of 3D reconstruction. The system aims to enhance the efficiency while maintaining the accuracy of the 3D reconstruction process by addressing key challenges including identifying high-utility views, sensor fusion for 3D perception, and view planning.

1.1 Motivation

This thesis focuses on the development of an intelligent data acquisition system for UAV-based 3D reconstruction to help mitigate the limitations and complexities associated with traditional UAV-based data acquisition systems. The primary motivation behind this study is to establish a strong connection between the visual features observed in 2D images and the corresponding 3D model of the target. By accomplishing this, the aim is to overcome the substantial computational costs typically involved in handling and processing large volumes of 3D data during UAV flight missions. This study proposes novel methodologies to improve the quality and robustness of UAV-based 3D reconstruction systems by focusing on three crucial steps of the data acquisition process: identifying high-utility views for multi-view 3D reconstruction, enhancing depth map quality as the input of the 3D reconstructor through a multi-task sensor fusion model, and appearance-based Next-Best-View (NBV) planning for 3D reconstruction.

Although the target task of this research is UAV-based imaging for 3D reconstruction of infrastructure, the developed methodologies can be applied to different domains where scanning a target is necessary. High-qaulity 3D models are not only used for condition assessment and monitoring of structures, but are also essential to the fields of virtual and augmented realty, game development, medical imaging, and education.

1.2 Objectives

The primary objective of this research is to design and implement an intelligent data acquisition system for UAVs. This system aims to enable UAVs to efficiently utilize their onboard sensors to accomplish several tasks. Firstly, it focuses on effectively

perceiving the reconstruction target through object detection and depth completion. Additionally, it aims to identify the most valuable views that contribute to the reconstruction process through utility assignment to views. Finally, the system encompasses the ability to strategically manipulate the UAV's path to reach the optimal viewpoints, thus facilitating Next-Best-View (NBV) planning. Each of these tasks have objectives of their own that, despite separability, are not independent. These objectives include:

1.2.1 Accuracy and Resource Efficiency Trade-off

In the context of 3D reconstruction (Chapter 2), the reconstructed model should be consistent with the ground truth model as a requirement for condition assessment carried out based on the results. However, factors such as limitations on flight time and energy consumption restrict the length of the trajectory along which data acquisition is being carried out. In Chapter 2, this is subjected to the proposed multi-view 3D reconstruction model by limiting the size of the input set of images while seeking high reconstruction accuracy. Consequently, the choice of which views to involve in the 3D reconstruction process will take care of the trade-off between accuracy and resource efficiency. Same objective is applied in Chapter 4 where views are actively chosen to be representative of the visual features from the corresponding range of viewpoints while avoiding views with redundant information.

In the context of Chapter 3, a different aspect of this trade-off is considered. Since depth completion and object detection are two crucial tasks often used for aerial 3D mapping, path planning, and collision avoidance of UAVs, performing both tasks while fusing the data from the camera and the depth sensor poses a challenge to resource efficiency. On the other hand, the accuracy of the detected targets in the scene as well as the generated depth maps impact the reconstruction process significantly.

As later used in the evaluation of the method proposed in Chapter 4, fusion of the resultant depth maps from Chapter 3 can be used to reconstruct the final 3D model. Failure in producing dense, accurate depth maps prevents develop and assess both reconstruction and view planning models. To address this challenge, the proposed multi-task model efficiently fuses depth and RGB data to produce high quality depth maps for 3D reconstruction.

1.2.2 Robustness

Employing reliable LiDAR sensors that produce high-resolution 3D point clouds is a viable option to enhance aerial photogrammetry. However, it is important to note that these sensors can introduce noise into the measurements, thereby reducing the confidence of decision-making algorithms. Additionally, using LiDAR sensors with higher performance guarantees may exceed the payload capabilities of drones and pose power consumption challenges. The solution developed in Chapter 3 needs to be robust to such noise and ensure stable depth maps.

1.2.3 Resilience

In the context of view planning, the method proposed in Chapter 4 must be robust to mistakenly generated viewpoints. In this regard, robustness of the system is demonstrated in the form of the system's ability to recover from sub-optimal or erroneous actions.

1.2.4 Execution Time

Algorithms designed to run onboard the drone, utilizing the available computational resources, need to exhibit timely execution. This requirement becomes particularly

crucial when safety-critical decisions rely on the output of these algorithms. Ensuring that the algorithms can run within acceptable time constraints is essential to enable real-time decision-making, thereby minimizing latency and potential risks. By efficiently utilizing computational resources, these onboard algorithms can provide timely and reliable outputs, contributing to the overall safety and effectiveness of the drone's operations.

In Chapter 3, this concept is considered by merging two crucial tasks of aerial depth completion and object detection. This not only avoids re-learning some features towards two closely related vision tasks during training, but also saves computation time during onboard inference when the jointly learned features in a single pass are utilized for two tasks instead of two passes through two separate networks. Additionally, the proposed method in Chapter 4 avoids carrying out any online reconstructions of the target and is based solely on visual inputs which reduces the computation time significantly.

1.3 Contributions

In this thesis, three essential steps of an intelligent data acquisition system for UAV-based 3D reconstruction are developed. This research mainly focuses on optimizing the UAV flight missions in terms of the travelled path and the use of onboard computation resources. The primary contributions of this work are listed below:

- **A pipeline for generating input sets of multi-view 3D reconstruction algorithms.** A feature matching-based pipeline is developed for the generation of input sets for multi-view 3D reconstruction models. The improvement of the reconstruction results is demonstrated on both a novel 3D reconstruction architecture and an existing model in the literature.

- **A novel multi-task approach to aerial depth completion and object detection for UAV-based imaging.** A novel approach is developed to simultaneously perform depth completion and object detection tasks in a single pass. The proposed method is based on an encoder-focused multi-task learning model that exposes the two tasks to jointly learned features. The study demonstrates how incorporating semantic expectations learned by the object detection pathway can enhance the performance of the depth completion pathway by inferring missing depth values in the scene. This step paves the way for the third step with identifying the target infrastructure and producing high-quality depth-fusion-based 3D reconstruction.

- **Bag-of-Views (BoV): A novel appearance-based computational representation of reconstruction target.** A novel approach is introduced to track and retrieve visual features of the reconstruction target. This concept is applied to both tasks of dataset filtering (offline view selection) and online Next-Best-View (NBV) planning.

- **A fully appearance-based approach to reinforcement learning of next-best-view planning for 3D reconstruction of infrastructure.** A novel reinforcement learning model based on the concept of Bag-of-Views is developed, training an agent to execute sequential view planning without tracking any partial or full reconstructions during training or inference time.

1.4 Thesis Outline

This thesis is structured to focus on three different components of an intelligent data acquisition system individually yet effectively.

Chapter 2 focuses on developing a novel computational model for knowledge representation of the reconstruction target.

Chapter 3 presents a multi-task approach for aerial depth completion and object detection.

Chapter 4 introduces a fully appearance-based reinforcement learning approach to NBV planning.

Each chapter details background, related work, methodology, experiments, results, and discussions for the corresponding component of this system. By addressing the challenges of view selection, sensor fusion, and view planning, this research aims to advance the field of UAV-based 3D reconstruction, enabling more accurate, efficient, and reliable reconstruction of 3D scenes while automating the process. The proposed methodologies and algorithms contribute to the development of intelligent data acquisition systems for UAV-based 3D reconstruction, with potential applications in various domains requiring high-fidelity 3D modeling. Figure 1.1 demonstrates the thesis outline with interactions between different components developed in each chapter.

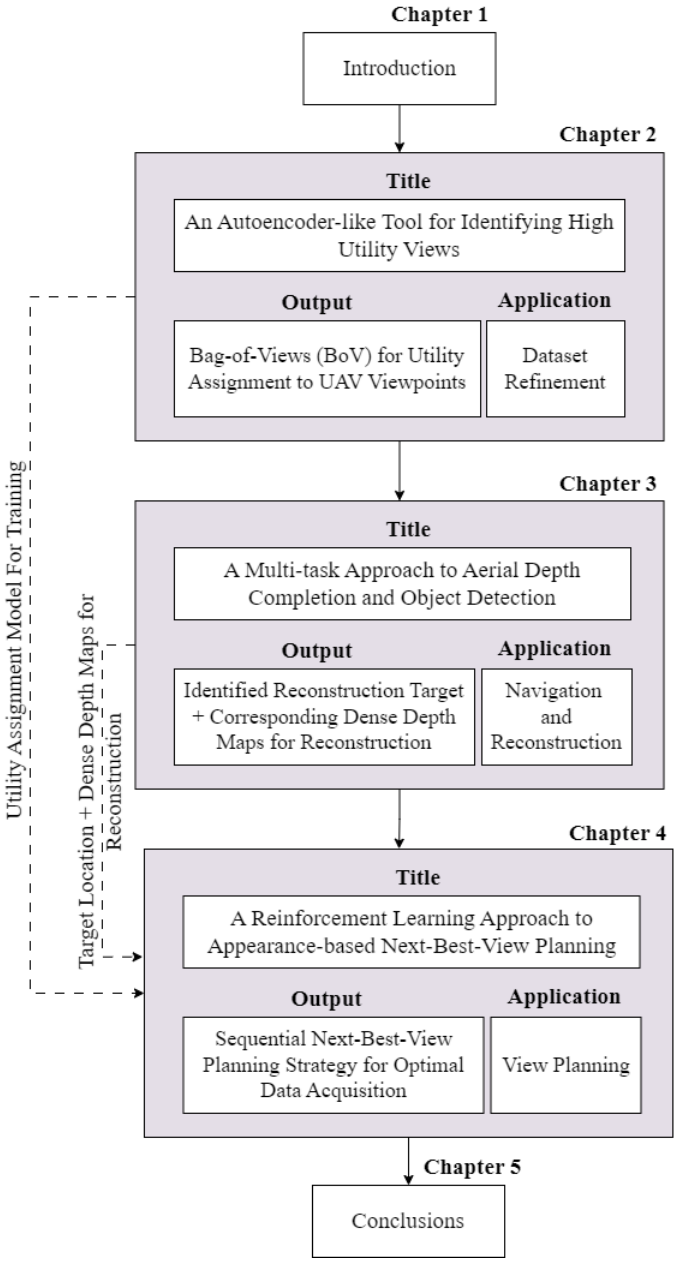

Figure 1.1: Thesis organization.

Chapter 2

Where to look: Identifying Better Views Through an Autoencoder-like Tool

2.1 Introduction

The main goal of image-based 3D model reconstruction is to utilize single or multiple images to infer the 3D shape of a target object. This well-studied problem has gained significant attention due to its applications in computer animation, virtual reality, aerial photogrammetry, and condition monitoring of structures and objects across various scales [6]. While humans excel at perceiving the overall 3D shape of objects through visual cues, the reconstruction of 3D models from 2D images remains a challenging and open problem.

With the growing interest in remote sensing and inspection, the field of 3D model reconstruction and mapping has been invested in more than ever before. Traditional methods in the field often face limitations such as difficulties in handling self-occlusion [7], large viewpoint distances [8], or the need for precise camera calibrations [9]. These methods also struggle with noisy and irrelevant data, requiring additional pre-processing steps [10].

As opposed to the traditional methods, deep learning-based methods can overcome these problems without the need to calibrate the camera and are robust to noisy

inputs [11, 12]. They are also capable of handling occlusion as they can learn to pay attention to holistic features of the set of images rather than local feature [13].

In the context of 3D reconstruction of structures and infrastructure, for efficient implementation, it is imperative to find a set of 2D images that does not compromise the 3D reconstruction quality. In addition, selecting the views that contribute the most to the 3D reconstruction process to feed into the reconstruction algorithm, instead of using the entire set of acquired data, improves resource efficiency in computation. The presented work describes a novel 3D reconstruction framework based on aerial 2D images acquired by simulating a UAV viewpoint in Blender [14] on different objects. The concept is motivated by the perceived ability of single-view reconstruction models to establish a utility for the input image.

The proposed framework consists of three parts: a sparse autoencoder-like model for 3D reconstruction from a single view, a selective sampling method for creating optimal view sets, and a variational autoencoder-like model for multi-view 3D reconstruction with the created view sets. It is proposed that in order to ensure high-quality models using multi-view 3D reconstructors, the views contained in the input set must necessarily satisfy two conditions. After this proof of concept using the autoencoder-like tool, this idea is generalized to non-deep learning-based methods and use it for dataset refinement. In this context, the Bag-of-Visual-Words [15, 16] is employed as a simplified vision model to represent the agent's knowledge of the target and introduce spatial information to the visual vocabularies to form a Bag-of-Views (BoV), a model to record and retrieve the visual features of the target from different viewpoints. First, experimental results demonstrate how the selection of the views using the proposed appearance-based heuristics affects the 3D reconstruction process and use the observations to refine already acquired datasets. Furthermore, the BoV model is applied to dataset refinement to show its efficacy in terms of identifying high-utility views and

reducing the number of images required for achieving high-quality reconstructions.

2.2 Background and Related Work

The problem of 3D model reconstruction has captivated researchers from diverse fields and is extensively studied. It holds significant importance due to its wide range of applications, including robotic motion planning and object manipulation.

In the realm of 3D object reconstruction, methods can be classified into two general approaches: classical methods and learning-based techniques. Classical methods, rooted in established algorithms and mathematical models, have long served as the foundation for addressing this challenge. On the other hand, learning-based techniques leverage the power of machine learning and deep learning, utilizing large-scale datasets and deep neural networks to reconstruct complex 3D models.

2.2.1 Traditional 3D Reconstruction Algorithms

Image-based 3D Reconstruction

Based on the number of images used for the algorithms in this category, these methods are classified into single- or multi-view reconstruction algorithms. These methods extract depth cues extracted from the change in the positioning of the visual cues of the objects in the scene as observed from multiple viewpoints at once (multi-ocular depth cues) or a single still viewpoint (monocular depth cues) [17, 18].

The process of extracting 3D structures from views captured by a moving camera around a scene is called Structure-from-Motion (SfM) and was first introduced by Longuet-Higgins [19]. This technique which is the fundamental component of many software applications [20] in the field including Agisoft [21], MeshLab [22], and CloudCompare [23], serves as the basis for many non-deep learning-based 3D reconstruction

methods. SfM leverages geometric constraints and estimates camera poses by the correspondences between visual features across images. Despite its effectiveness, SfM has limited handling of scene ambiguities such as regions not rich in texture. Lack of texture can mislead the feature extraction algorithm and lead to poor results [24]. SfM's dependency on feature extraction also limits its scalability to large scenes due to the expensive computation and memory cost [25]. It is also restricted by dynamic lighting conditions, sparsity in camera poses, and self-occlusion.

Depth-based 3D Reconstruction

With the recent advances of depth sensors, depth images are gaining more attention for 3D reconstruction. Methods that use depth images for 3D reconstruction are classified as depth-based methods, as opposed to the aforementioned image-based methods. This process includes five main steps depending on the intended output format. These steps are as follows [26]:

I. **Depth Map Denoising and Enhancement**: In this step, the noise and invalid depth values are filtered using mean, median, or Gaussian filtering. Other advanced depth map denoising and enhancement methods are discussed in Chapter 3.

II. **Point Cloud Computing**: Every pixel in the depth image corresponds to the distance between a point on the surface of the target object and the depth sensor. Knowing the location of the camera in the global coordinate system of the scene and the camera parameters, the location of the mentioned point can be calculated.

    First, the camera intrinsic matrix is obtained in the form below. The camera intrinsic matrix is a 3×3 matrix that describes the mapping between 3D world

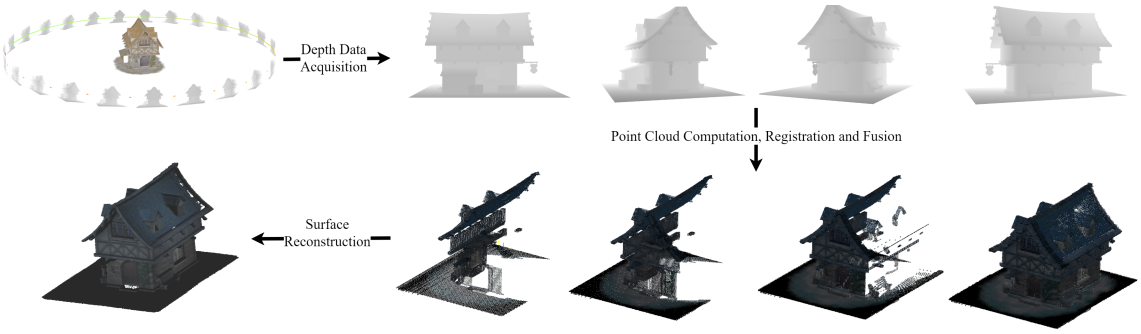

Figure 2.1: Visualization of the point cloud generation process from depth maps.

coordinates and the 2D image coordinates.

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

In this equation, c_x and c_y specify the optical center offset, s is the axis skew which is normally zero, and f_x and f_y determine the focal length in the x and y directions. The below equation can be written to relate the image pixel coordinate system with that of the 3D world [27].

$$z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} R|T \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.2}$$

Here, the target point's location with respect to the depth image coordinate frame is denoted by (u, v) which is to be mapped to x and y with depth z. R denotes the rotation matrix and t represents the translation vector. $\begin{bmatrix} R|T \end{bmatrix}$ refers to the concatenation of R and T.

III. **Registration:** A rigid transformation is needed to ensure the obtained point clouds from the previous step are based on a unified coordinate system. One of the most popular registration methods is the Iterative Closest Point (ICP) algorithm [28]. The core idea behind this algorithm is to iteratively refine the transformation needed to align a target point cloud to a reference point cloud. The algorithm tries to find the optimal rigid transformation including the rotation and translation matrices that minimize the distance between the two point clouds.

IV. **Fusion:** After registering the point clouds generated from the depth maps, the resultant point cloud will have redundancies and inconsistencies due to the faults in the sensor or capturing process or environmental conditions. This can also be caused by the imperfection of the registration algorithm in the previous step. As a result, the values from the aligned depth maps must be efficiently fused to generate a consistent point cloud. This fusion can be achieved using various methods, such as selecting the minimum or maximum depth value per voxel or applying weighted averaging based on confidence measures or depth quality. One method is to assign to each voxel its distance to the reconstructed surface [29]. This value is called Signed Distance Field (SDF) value and determines whether the voxel is in front or behind the surface. For efficiency in computing, [30] proposed to only consider the voxels closer to the surface. Their Truncated Signed Distance Function (TSDF) solves the memory and computation burden of calculating and storing the SDF values for all voxels. This method is used throughout this thesis to evaluate the proposed dataset refinement and view planning technique.

V. **Surface Reconstruction and Post-processing** After the fusion of all the point clouds, a surface reconstruction algorithm such as marching cubes [31]

can be used to generate a surface representation of the scene.

The overall process is demonstrated in Figure A.2.

2.2.2 Deep Learning-based 3D Reconstruction

To overcome the mentioned drawbacks faced by traditional 3D reconstruction algorithms, learning-based methods were proposed, often utilizing autoencoder-like neural network models [32]. Depending on the target object and the type of the task, these methods can accomplish single- or multi-view 3D reconstruction [33], [34]. Unlike the traditional 3D reconstruction methods, deep learning-based methods often need less images than the traditional methods and require no known camera calibration [35].

Single-view 3D Reconstruction

Deep learning models developed to carry out single-view 3D reconstruction mostly follow an autoencoder-like architecture which accepts as input one image of a target and generates a 3D representation of the target. Such models are usually trained on 2D single images paired with their corresponding 3D model [36] or without direct supervision from a ground truth 3D model [37] where a secondary representation of the target is learned by the model. Due to the ambiguities risen from lack of information about the appearance of the model from more than one point of view, this task is considered to be ill-posed and of less applicability and scalability to high-accuracy reconstruction for condition monitoring of large structures. However, this task remains essential for quality assessment of industrial products [38].

Multi-view 3D Reconstruction

Employing single-view image for 3D reconstruction has many challenges such as suffering from self-occlusion and a lack of enough information from other viewpoints.

Thus, many studies in literature focused on multi-view reconstruction. For instance, in [39] a new structure namely projective generative adversarial networks (PrGANs) is presented. This technique attempts to match projections of 3D objects with 2D views by training a deep generative model. In [40], a convolutional deep belief network is proposed to learn the probability of each voxel employing prior knowledge to fill in the unknown voxels to complete the reconstruction. Convolutional autoencoder network is another structure which is used for 3D reconstruction purpose. In [1], a 3D convolutional deep belief network is trained to generate volumeteric representation of an object from a one single depth map. [41] present a convolutional network which is able to produce 3D representation of an unseen object with a single image. In this research, the network can produce an RGB image and a depth map of the object as seen from an arbitrary view. By fusing these different views a full point cloud of the object can be generated. Moreover, in [42], an end-to-end framework based on 3D interpreter network is proposed to estimate 2D keypoint heat-maps and the 3D object structure. This research attempts to complete 2D keypoint estimation and 3D structure and viewpoint recovery simultaneously.

2.2.3 Bag of Visual Words

The Bag of Visual Words (BoVW) model is inspired by the Bag of Words (BoW) model used in natural language processing and both are variants of the Bag of Features method. To simply explain the methods, with a BoW model a document is represented as a bag of words, where the order of the words does not matter, but their frequency of occurrence does. The frequency of the occurrence of the words can be used for sentiment analysis of the sentences they are used in [43]. Similarly, in the BoVW model, an image is represented by its lower-level features as a bag of visual words where the frequency of certain detected features helps with understanding the

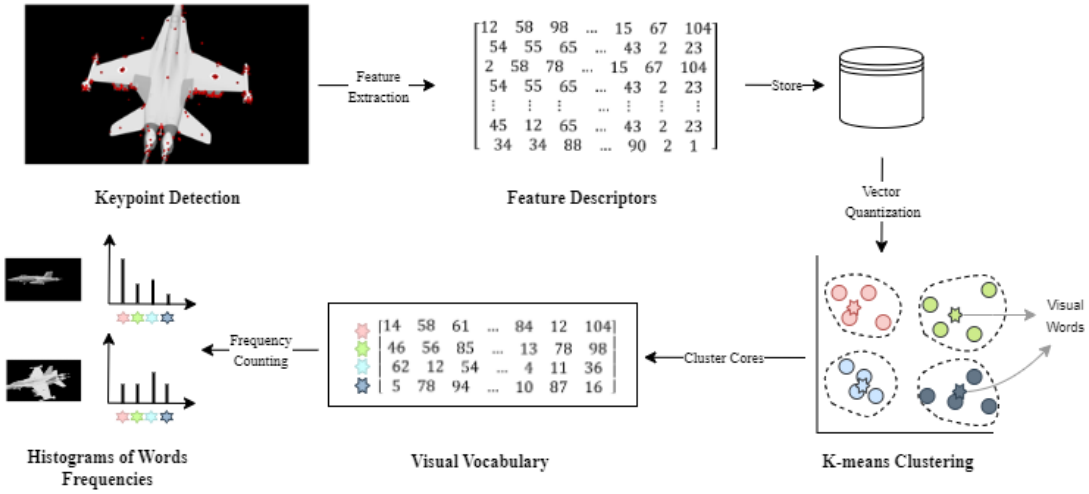

Figure 2.2: Simplified Visualization of the Bag-of-Views model. The feature descriptors of each view are assessed and update only the vocabulary that belongs to their corresponding range of views.

image [15]. The BoVW model involves the following steps [44]:

i. Feature Extraction: Feature extraction refers to the process of detecting points of interest, e.g. salient regions, on an image and quantifying the features of such points through what is called a feature descriptor. Feature extraction can be carried out in a variety of ways such as detecting keypoints and extracting Scale-Invariant Feature Transform (SIFT) features [45], Speeded-Up Robust Features (SURF) [46], ORB features [47], or applying a grid at regularly spaced intervals as in Dense keypoint detection [48]. In this manner, every image will be presented by a set of standard descriptors

ii. Visual Vocabulary Generation: The local features extracted from multiple images in the last step are clustered using a clustering algorithm. Generally, K-means [49] which is based on vector quantization [50] is used to generate K cluster cores, i.e. visual words, where K is originally set manually.

iii. Quantization: Each feature descriptor is assigned to its closest visual word and the distance between the two is calculated using a metric such as Euclidean

distance or cosine similarity.

iv. Histogram Creation: In classification and retrieval tasks, this step corresponds to counting the occurrence of each generated visual word from the learned vocabulary and creating a histogram of word frequencies.

Figure 2.2 demonstrates the aforementioned steps.

2.3 Methodology

This study aims to create an assessment tool capable of evaluating the contribution of each image to the reconstruction process. To achieve this goals, it is necessary to bridge the 2D image features to the 3D model of the target. For this purpose, a multi-view reconstruction tool is required that enables us to assess the functionality of different sets of views.

In this section, first, a novel variational autoencoder-like tool is proposed for multi-view 3D reconstruction. Furthermore, a two-part tool is introduced that enables utility assignment to the views based on their individual and group performance in reconstruction. Then, this technique is extended to build a computational representation of the target that can help with selecting the views without any reference to a partial or full reconstruction of the target which is later used for dataset refinment.

2.3.1 Multi-view 3D Reconstruction

The multi-view 3D reconstruction network is a Variational Autoencoder (VAE) which takes a set of ten 2D grayscale images, each of size 32×32 as input and outputs a $64 \times 64 \times 64$ voxel grid.

Encoder: The encoder of the network predicts the distributions of 128 features by estimating two vectors of dimension 128×1 corresponding to the means and the

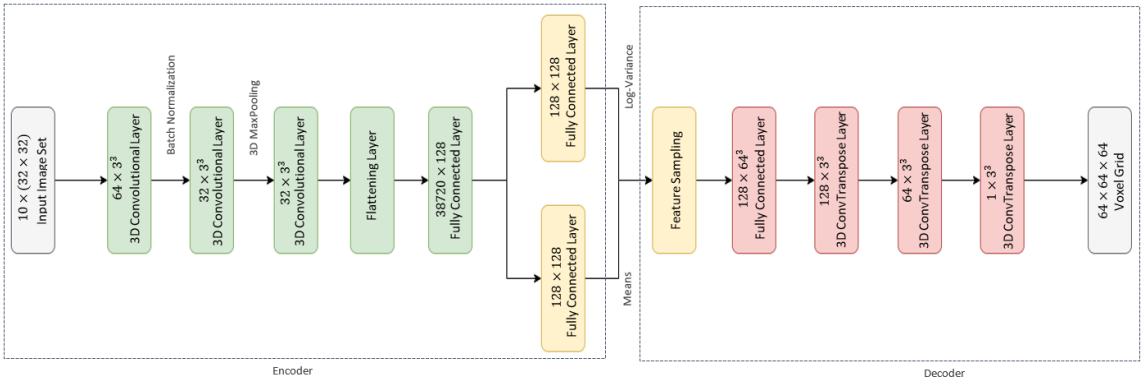

Figure 2.3: Detailed architecture of the proposed variational autoencoder-based 3D model reconstructor.

logarithm of variances of the features. The architecture involves a 3D convolutional layer followed by a batch normalization layer, a 3D convolutional layer, a max pooling layer, another 3D convolutional layer and a flattening layer which is then mapped to a vector of the size of the feature space. The max pooling layer has a kernel size of $1 \times 3 \times 3$. The three convolutional layers have a kernel size of 3×3 each. There are 64, 32, and 32 channels respectively for each of the convolutional layers.

Decoder: The decoder is used to map the extracted samples of the feature space to the voxel grid output. The vector of feature samples is first reshaped to the target grid size of $64 \times 64 \times 64$. Three consecutive 3D transposed convolutional layers with 128, 64, and 1 channels and kernel sizes of $3 \times 3 \times 3$ follow the reshaping layer. The activation function of the last convolutional layer is set to sigmoid which produces a probability estimate for whether the respective voxels are occupied or not.

Loss Function: The function estimating the network's loss consists of two terms: the reconstruction loss and the Kullback-Leibler divergence loss. The reconstruction loss takes into account the difference between the ground truth and the reconstructed volumetric shapes utilizing the binary cross-entropy function. This term is defined as below:

$$l_1 = \frac{1}{N}\sum_{i=1}^{N}[y_i log(p_i) + (1 - y_i) log(1 - p_i)] \tag{2.3}$$

where p_i is the output of the network and y_i is the ground truth grid. The KL-divergence term in the loss function acts as a regularizer for the probability distributions in the feature space. This loss compares the feature vector with a probability distribution of mean 0 and standard deviation 1. This will prevent the network from producing sparse feature distributions which makes it possible for the network to interpolate between classes while encountering features of different classes.

$$l_2 = \sum_{i=1}^{N}[\sigma_i^2 + \mu_i^2 - log(\sigma_i) - 1] \tag{2.4}$$

where σ_i and μ_i are the mean and variance of the ith feature's probability distribution in the feature space.

Note that the 3D matrices which go through these formulas are flattened. That is why only one index is used to represent iterations.

2.3.2 A Framework for Identifying Best Set of Views

The proposed VAE requires a set of input images for 3D reconstruction, which is often sampled randomly from the dataset. The quality of reconstruction from such images cannot be guaranteed through random sampling, given the non-uniform distribution of features within the images. In this section, the best views identified from the single-view 3D reconstruction network output are utilized to find the optimal set of input images for the VAE. Intuitively, the best set of images leading to a high-quality reconstruction should encompass all possible features given a target object. Random sampling can lead to inefficient reconstructions, warranting multiple iterations until a satisfactory reconstruction is reached. This can be understood with the following

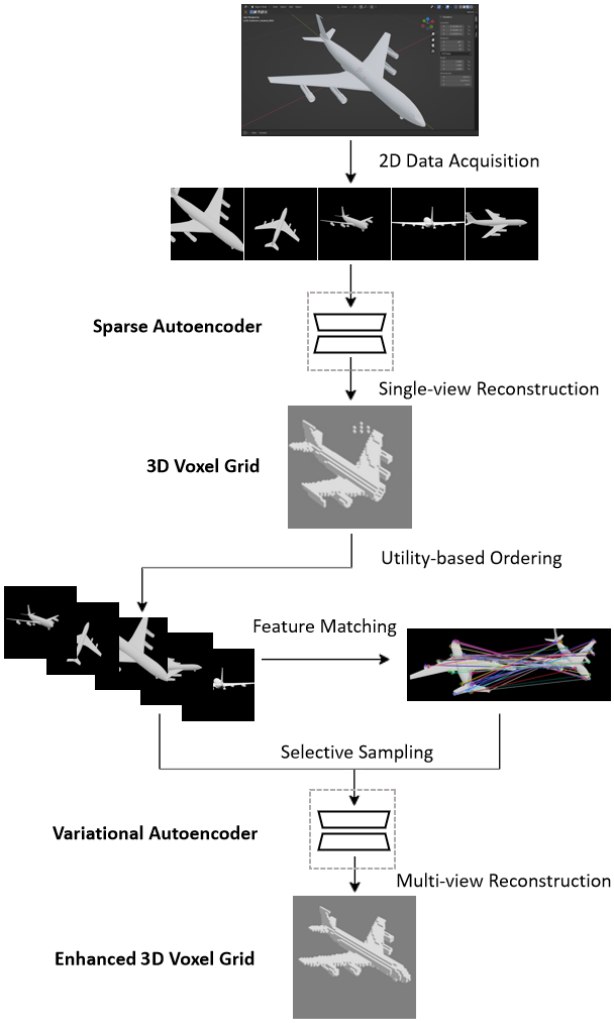

Figure 2.4: Proposed framework for enhanced 3D reconstruction using single- and multi-view autoencoders.

insights:

- The randomly sampled images can present similar features of the target object. To ensure maximum features captured within the set and a uniform scanning of the target object, it is imperative to sample distinct images.
- To further optimize the information held in the image set, images that capture the most informative views should be selected.

To alleviate the challenges presented by the above insights, this work presents

the process of selective sampling that encapsulates maximum information through varying viewpoints in the VAE input set. It is assumed that the views that correspond to the best reconstructions with the SAE encapsulates the most useful information required for reconstruction. The same views can be utilized to generate an enhanced reconstruction using the VAE, while ensuring a set of distinct high-performing images. The corresponding inputs to the SAE are sorted with respect to their reconstruction accuracies (IoU). The sorted images, referred to as a utility-ordered set, are then used for a selective sampling process that finds the most informative views with the least redundant features. The selective sampling process ensures that the images existing in each set are distinct, referring to different sides of the target object. The process initiates by considering the view leading to the highest IoU from the SAE output as the best image to put into the set. Following images are then processed in a descending order of IoU. Each view is checked against the preexisting views in the set using OpenCV ORB feature matching [51]. Compared to other methods, such as SIFT [45], this feature matching method has proven to work more efficiently when the orientation of the image is the main concern of the comparison made between two views. An image is selected as a part of the set if the number of matching features between the view in question and the existing views is lower than a predefined threshold. The entire proposed framework is shown in Figure 2.4.

2.3.3 Bag of Views: Appearance-based Approach for Selecting Best Views

In this section, a computational representation of the views is introduced in terms of the visual features of the scene captured from the respective viewpoints. As discussed in [52], for a successful multi-view 3D reconstruction, two key conditions must be satisfied:

- The views in the set must present features of the target that are distinct from the ones presented by other views in the same set,
- Each view by itself must be rich in the number of visual features it is revealing of the target.

The i^{th} view in the set is denoted set as χ_i. Each view χ_i is encoded to and is represented by its extracted features using a feature extracting algorithm such as Scale-Invariant Feature Transform (SIFT) [45]. Thus, χ_i will be a 2D matrix with dimensions $m \times n$ where m is the number of extracted features and n is the number of values in the feature descriptors. In the case of SIFT, each row of this view matrix represents a 128-dimensional feature descriptor where each element represents a certain attribute of the local feature detected in the image patch. Each view can be denoted by iterating over its resulting feature descriptors $\chi_i(j,:) = \{f(j,k) : k \in \{1,2,...,n\}\}$ where $f(j,k)$ is the k^{th} value in the j^{th} descriptor of the image. These conditions for a set of views result in a greater distance between corresponding descriptors from two view representations denoted as $dist(\chi_i(j,:), \chi_{i+1}(j':))$ for consecutively selected random views χ_i and χ_{i+1}. A greater distance ensures a better reconstruction quality for a limited-length trajectory. In this study, the cosine distance metric is used to measure the dissimilarity between the descriptors and the visual words since its consistent range of outputs allows for easy interpretation and score comparison. Thus,

$$dist(\chi_i(j,:), \chi_{i+1}(j':)) = \cos\left(\chi_i(j,:), \chi_{i+1}(j':)\right) \tag{2.5}$$

where $j \in \{1,2,...,m\}$ and $j' \in \{1,2,...,m'\}$ with m and m' being the number of representative descriptors of χ_i and χ_{i+1}:

$$\cos\left(\chi_i(j,:), \chi_{i+1}(j':)(j)\right) = \frac{\langle \chi_i(j,:),\ \chi_{i+1}(j':)\rangle}{|\chi_i(j,:)|.|\chi_{i+1}(j':)|} \tag{2.6}$$

Since the extracted features belonging to a target are prone to self-similarity, the feature descriptors of different regions of the target can be clustered based on their similarity and, instead of pair-wise comparison of all feature descriptors belonging to all views in the set, they can be compared to the cluster cores. In addition, the necessity of there being a model-free view planning with no pre-training requires learning these cluster cores iteratively from the incoming information. This clustering of the visual features is inspired by [53] where cluster centers of the quantified feature descriptors were used to form Bag-of-Words. This method was later applied to a variety of computer vision tasks from loop-closure detection in visual Simultaneous Localization and Mapping (vSLAM) [54, 55] to appearance-based navigation [56].

In the context of view planning for a single target with one or more symmetry axes, learning a global visual vocabulary for the entire model can be prone to overconfidence in recognizing certain visual words. Therefore, a method is proposed to track visual features of the target captured from different viewpoints using distinguished visual vocabularies for different regions, incorporated within what is referred to as a Bag-of-Views (BoV). As well as mitigating the symmetry challenge, the computation cost will be significantly less with the viewpoint-based vocabularies; a smaller number of visual words is required to describe a portion of the target rather than all parts of it and the new view will only update its corresponding vocabulary among the BoV. Below are the steps involved: Figure 2.5 visualizes the workflow for creating a BoV for a sample object from the ModelNet40 [57] dataset. Below are the steps included in this workflow:

1. **Feature Extraction:** Using a feature detection algorithm, in the case of this study SIFT, local features of the captured image at position T are extracted in the form of 128-dimensional vectors. T is the position of the camera in the Cartesian coordinate system with its origin located at the center of the scene.

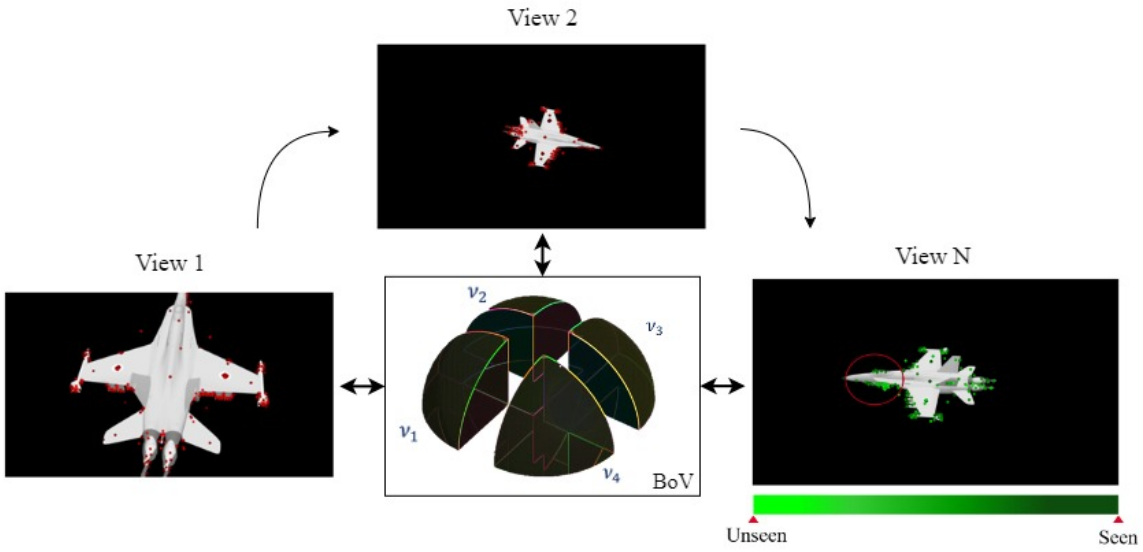

Figure 2.5: Simplified Visualization of the Bag-of-Views model. The feature descriptors of each view are assessed and update only the vocabulary that belongs to their corresponding range of views.

2. **Utility Assignment:** Depending on the application at hand, assigning utilities to the views can be divided into two different cases:

   (a) In the case of dataset refinement where the views have already been captured, a decision should be made about including each view in the input set of the reconstruction algorithm based on its utility. The question simply is "does this new view help the reconstruction process?". To answer that, the extent to which this view satisfies the two conditions mentioned before is examined. Each of the feature descriptors from the previous step are compared with their closest visual word through vector quantization [50]. Then, a distance metric is used to measure the dissimilarity between feature descriptors and the supposed visual word that would represent them in the corresponding vocabulary in the BoV, denoted as ν_T. This process is repeated for all of the descriptors and the sum of the dissimilarity scores

is used to decide whether to ignore or utilize the view in the reconstruction process. If the final score is a non-zero positive value, it is included in the set and proceeds to the third step, otherwise it is ignored.

(b) In the second case, the BoV model is utilized to train a reinforcement learning agent in proposing Next-Best-Views (NBVs) based on the appearance of consecutively captured views. Further elaboration on the learning process will be provided in Section 4.3.1. In this section, the focus lies primarily on the utilization of this model to shape the reward function within the specified context. During the training of the reinforcement learning agent, the change in the corresponding vocabulary of the Bag-of-Views (BoV) after capturing a new view at location T is used to shape the reward function. The higher difference between the new and previous BoV implies that the new view contains more unseen features and results in higher rewards.

3. **View Representation:** This step is a continuation of case I. Depending on the position and orientation of the captured viewpoint, the extracted features update the specific vocabulary assigned to the range of views that the new view belongs to. This updating includes clustering of the descriptors belonging to that region using a clustering algorithm such as K-means. Thus, every group of the cluster centers in the BoV, namely every regional vocabulary, describes the appearance of the target from viewpoints that are close in position and orientation. A simplified visualization is demonstrated in Figure 2.5.

Algorithm 1 showcases the pseudo-code for the creating a BoV model.

Algorithm 1 Bag-of-Views Model for Offline View Selection

Require: Number of view ranges N, Number of words W, Database $\mathcal{D}$

Initialize BoV$\{\nu_{1:N}\}$, Feature extractor $\mathcal{F}$
while There is data to process in $\mathcal{D}$ **do**
  $T, X \leftarrow$ Load data from $\mathcal{D}$
  $id_{\text{view}} \leftarrow T.\text{azimuth}//\frac{2\pi}{N}$
  $\chi \leftarrow \mathcal{F}(X_i)$
  $dist \leftarrow 0$
  for $j = 1$ to W **do**
    $C \leftarrow \arg\max_k (\cos(\chi(j), \nu_{id}(k)))$
    $dist \leftarrow dist + (1 - 2 \times \cos(\chi(j), \nu_{id}(C)))$
    if $dist > 0$ **then**
      Update descriptors with χ for $\nu_{id_{\text{view}}}$
      $\nu_{id_{\text{view}}}.codebook \leftarrow$ Perform K-means on $\nu_{id_{\text{view}}}$
    else
      Remove X from $\mathcal{D}$

2.4 Implementation Details and Experiments

2.4.1 Evaluation Metric

Evaluating Voxel Grid Reconstruction

The **Intersection over Union (IoU)** metric is used to evaluate the quality of the 3D reconstructions. IoU reports the ratio of the correctly predicted occupied voxels from the network output and the volume of the full ground truth grid, shown in Equation 2.7:

$$IoU = \frac{\sum_{i=1}^{N} \mathbf{I}(p_i > \theta)\mathbf{I}(y_i)}{\sum_{i=1}^{N} \mathbf{I}(p_i > \theta) + \mathbf{I}(y_i)} \tag{2.7}$$

where θ binarizes the grids' occupancy probabilities and $\mathbf{I}(\cdot)$ is the indicator function.

Evaluating Point Cloud Reconstruction

Chamfer Discrepancy measures the distance between each point in one point cloud and its nearest neighbor in the reference point cloud bidirectionally. Different variants

of the Chamfer discrepancy might present the average or sum of these distances. Given the reconstructed point cloud R and the reference point cloud P, the Chamfer discrepancy is defined as:

$$d_{CD}(R,P) = \frac{1}{|R|}\sum_{r\in R}\min_{p\in P}||r-p|| + \frac{1}{|P|}\sum_{p\in P}\min_{r\in R}||r-p|| \tag{2.8}$$

Hausdorff Distance is a metric used to quantify the maximum of Euclidean distances between one point in a point cloud and its closest point in another cloud. The reported distance represents the largest separation between corresponding points in two point clouds. Assuming the reconstructed point cloud R and the reference point cloud P, the Hausdorff distance is defined as:

$$d_{HD}(R,P) = \max\left\{\sup\left[\inf\left(d(R,P)\right)\right], \sup\left[\inf\left(d(P,R)\right)\right]\right\} \tag{2.9}$$

While Chamfer discrepancy provides insight about the overall similarity or dissimilarity between two sets of points, the Hausdorff distance highlights the most extreme differences between the point clouds and focuses on the largest observed distance. When comparing the scanning results using these two metrics, a greater Chamfer discrepancy is interpreted as indicating poor overall coverage of the target, while a higher Hausdorff distance is associated with missed areas in scanning, resulting in holes in the reconstruction.

2.4.2 Dataset

Objects from 9 different classes in Princeton ModelNet dataset [1] are used for this experiment. Samples of this dataset are shown in Figure 2.6. Two key aspects were considered during the generation of the dataset to ensure that the views best represent those captured by an inspection drone:

- There is a light source in the environment creating unique shades on the objects.
- The objects are considered to be lying on the ground. In this manner, no views are available from the bottom of the object.
- The views are captured from different zooming scales and at randomly generated positions around the objects.
- The dataset contains outliers which are basically extra-zoomed in views. Including these outliers in the dataset makes the networks robust to noise and irrelevant views.

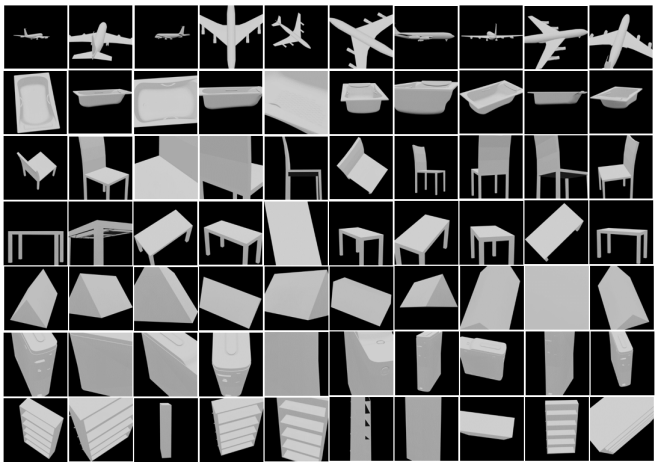

Figure 2.6: Samples from sets of views created from Princeton ModelNet [1].

2.4.3 Identifying Best Views for Multi-view 3D Reconstruction

As the intention is to apply this work in the field of aerial photogrammetry, a concept was proposed in Section 2.3.2 for the problem of trajectory generation for the viewpoints of the drone for a better 3D model Reconstruction. In traditional methods, it is necessary to maintain an approximately 80% overlap between the views so that

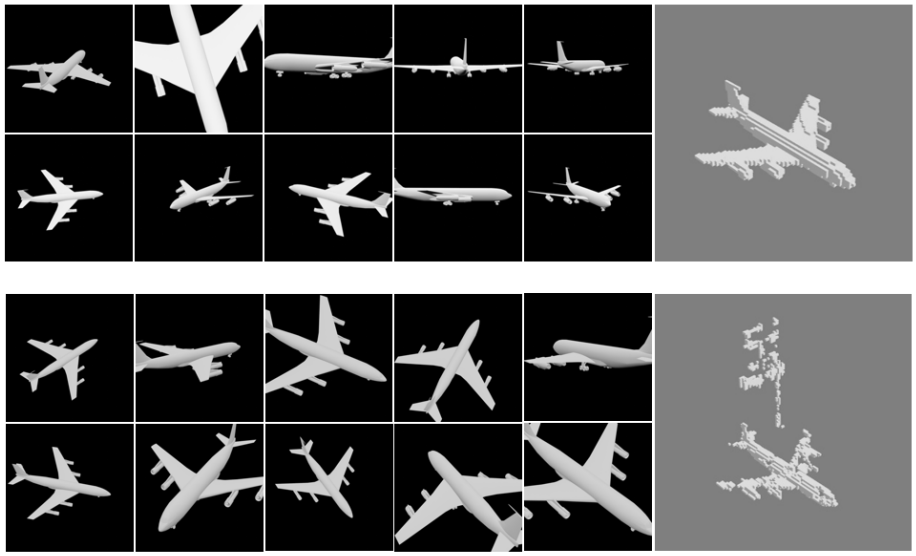

Figure 2.7: VAE output reconstructions using selective (top) and random (bottom) image samples for multi-view 3D reconstruction.

the alogrithm does not fail. However, it has been observed that the more distinction in the views leads to better reconstruction quality in the case of having an input set of limited length. On the other hand, more complete coverage of the object being inspected leads to better reconstructions. Examples of random and selective input sets as well as their corresponding reconstruction results are shown in Figure 2.7. The reconstruction obtained via selective sampling generates significantly better results when compared to random sampling.

2.4.4 3D Reconstruction Using Synthetic Images

A comparison was made between the results of the VAE multi-view 3D reconstruction network and the RNN-based 3D reconstructor proposed in [58]. The choice of this comparison reference is due to the architecture similarity of the two works, the main difference being the type of layers used to encode the input views into the latent space. Instead of using Long Short-Term Memory network to find the relativity of the visual

features in the views, 3D convolutional layers were utilized to collectively map the features of the entire set at once. The comparison of the results for networks trained on the sample classes from the ModelNet dataset are demonstrated qualitatively in Figure 2.8 and quantitatively in Table 2.1. As can be seen in the figures, both networks are capable of retrieving the geometric shape of the objects, but the VAE integrated with the input view selection method has exceeded the 3D-R2N2 network in most cases in terms of reconstruction quality.

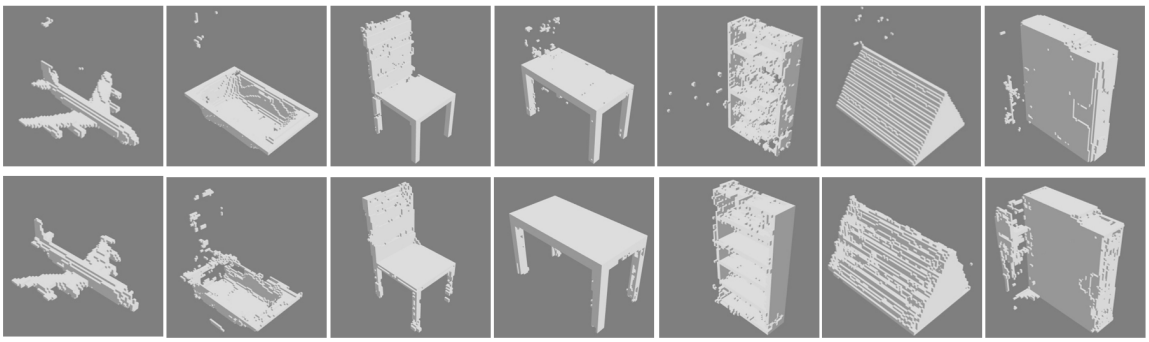

Figure 2.8: 3D shape retrieval result using synthetic images. The first row corresponds to the results of the VAE 3D reconstructor and the second row corresponds to the results of the RNN-based 3D reconstructor. Views are captured from the objects provided in the Princeton ModelNet dataset. Objects from left to right: airplane, bathtub, chair, table, bookshelf, tent, and Xbox.

Table 2.1: IOU values for each object reconstruction.

Object	3D-R2N2	Proposed VAE + Selective Sampling
Airplane	72%	79%
Bathtub	62%	73%
Bookshelf	81%	69%
Chair	71%	68%
Table	83%	88%
Tent	71%	92%
Xbox	68%	84%

2.4.5 Bag-of-Views for Dataset Refinement

To investigate the impact of the BoV size, the model's performance is evaluated by filtering two generated datasets comprising 288 views each, which uniformly covered two buildings from a selected dataset. Algorithm 1 was used to reduce the dataset size and evaluate reconstruction performance resulting from the remaining views. The two effective variables in this study are the number of view ranges defining the size of BoV and the number of visual words in each vocabulary of the BoV. The reconstruction results are compared based on their Hausdorff distance and Chamfer discrepancy with the 3D point cloud reconstructed using the original dataset of 288 views. The results are listed in Table 2.2. In addition to this quantitative comparison, the distance between the mesh reconstructions and those produced using the original dataset is visualized in Figure 2.9.

This analysis explores the question of achieving optimal performance by balancing model efficiency (number of selected views) and reconstruction quality. Shown in Figure 2.9, a larger BoV, achieved by increasing the number of vocabularies, results in a reconstruction that exhibits reduced spatial sparsity in the error surrounding the model. This interprets as a more uniform scan of the target when seeking to update visual vocabularies that are defined for a smaller view range. This means that for a fixed number of words in each vocabulary, each view has less opponents to be compared with and the resulting visual words are more local to that region. While each view has a higher chance to represent a certain view range in the vocabulary, there is a lower chance for its similar views to be accepted into the set, as the dominant local features have already been identified. This effect is reinforced when the number of visual words per each vocabulary in the BoV is increased. The higher the number of visual words attributed to each view range, the higher the chance of familiarity of the newly captured view and details exposed to it. For high-resolution samples of the

individual images comprising the composite figure shown in Figure 2.9, please refer to Appendix A.

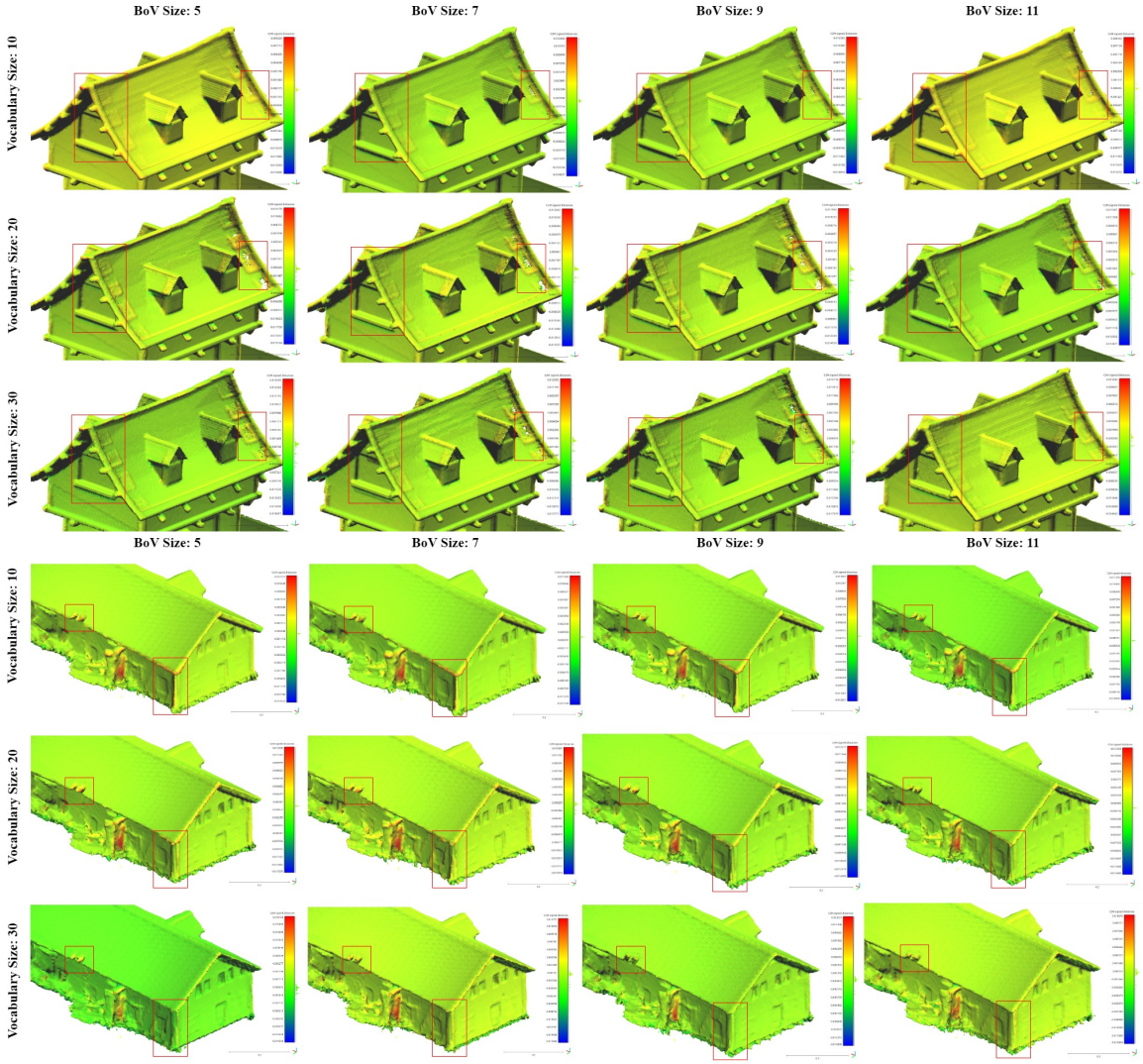

Figure 2.9: Studying the effect of the size of BoV and the vocabularies within it. From a qualitative perspective, it can been seen that increasing the size of BoV lowers the error sparsity while increasing the vocabulary size reduces the error values.

Table 2.2: Studying the influence of the size of Bag-of-Views on the dataset size and the reconstruction results from the TSDF fusion of RGB and Depth images

Size of BoV		5	5	5	7	7	7	9	9	9	11	11	11
Number of Words per View Range		10	20	30	10	20	30	10	20	30	10	20	30
Medieval Tavern	Number of Selected Views	143	67	37	142	63	34	140	62	35	146	63	42
	Hausdorff Distance (cm)	2.1509	1.8134	1.8901	1.8045	2.3907	2.2756	1.8803	2.3693	1.5216	1.4825	1.7841	1.4355
	Chamfer Discrepancy (cm)	0.1183	0.1935	0.4542	0.1206	0.4345	0.3742	0.1254	0.1906	0.2998	0.1109	0.1953	0.2979
Forest Ruin House	Number of Selected Views	102	38	23	92	33	24	98	42	32	99	47	36
	Hausdorff Distance (cm)	2.5810	2.2692	2.5231	2.0577	2.1886	3.1446	2.1244	2.2862	2.3698	2.04848	2.6412	1.7054
	Chamfer Discrepancy (cm)	0.2374	0.4554	0.6548	0.2416	0.5375	0.5775	0.1869	0.6057	0.5842	0.3115	0.5359	0.5535

2.5 Summary and Conclusion

This chapter tackled the process of 3D reconstruction from 2D images. An autoencoder-like tool was developed that, in addition to successful reconstruction results in the form of 3D voxel grids, was used to assign utility to each view of the target based on its contribution to the reconstruction process. In this framework, the reconstruction quality of a single-view autoencoder-like 3D reconstructor was used as an indicator of the utility of each view by itself. Then, a multi-view variational autoencoder-like 3D reconstructor was used to assess how much each view is contributing to the process as part of a whole set. This was done by employing a feature-matching method to evaluate the amount of relative visual information in the views that are ordered in terms of their IoU-based utility. This led to taking into account only the best views among a large set of views whose contained visual information does not cause redundancy and thus inefficiency in the input set of the reconstruction model. Building up on this concept, a novel appearance-based computational representation of reconstruction targets was introduced that can be of utmost utility for UAV-based aerial photogrammetry. The presented model enables utility assignment to the views by eliminating the need to track a full or partial reconstruction of the target. Instead, this approach involves tracking unfamiliar visual features using visual vocabularies encapsulated within what was referred to as a Bag-of-Views (BoV). Through experimenting the effect of different sizes of BoV and the vocabularies within it on reconstruction quality and the number of selected views, it was found that the BoV model achieved a remarkable reduction of views used for reconstruction (70.6% decrease) while simultaneously reducing the reconstruction error (33.5% decrease). In these experiments, the reconstruction results from a complete coverage scan of the target were used as baseline. These outcomes showcased the efficacy of the presented model in identifying optimal views for reconstruction.

Chapter 3

How to see better: Efficient Aerial Depth Completion and Object Detection Through a Multi-Task Network

3.1 Introduction

3D mapping and understanding of scenes are crucial tasks for Uncrewed Aerial Vehicles (UAVs) that are responsible for large-scale inspections and condition assessment. While visually inspecting a target such as infrastructure for monitoring or 3D modelling purposes, it is crucial for the UAV to efficiently deploy its onboard sensors to acquire information from its surroundings to ensure a safe, collision-free travel while satisfying the mission objectives.

Other than algorithms that are based on Structure-from-Motion (SfM) [59], using reliable LiDAR sensors that produce high-resolution 3D point clouds is an option. However, their measurements are typically prone to noise, thereby diminishing the confidence of decision-making algorithms. On the other hand, LiDAR sensors with higher performance guarantees may surpass the drones' payload capabilities and bring about power consumption problems. Accordingly, numerous techniques have been devised to address the challenges caused by defects and output sparsity of the sensors that the drones could carry. Among these techniques, depth completion methods

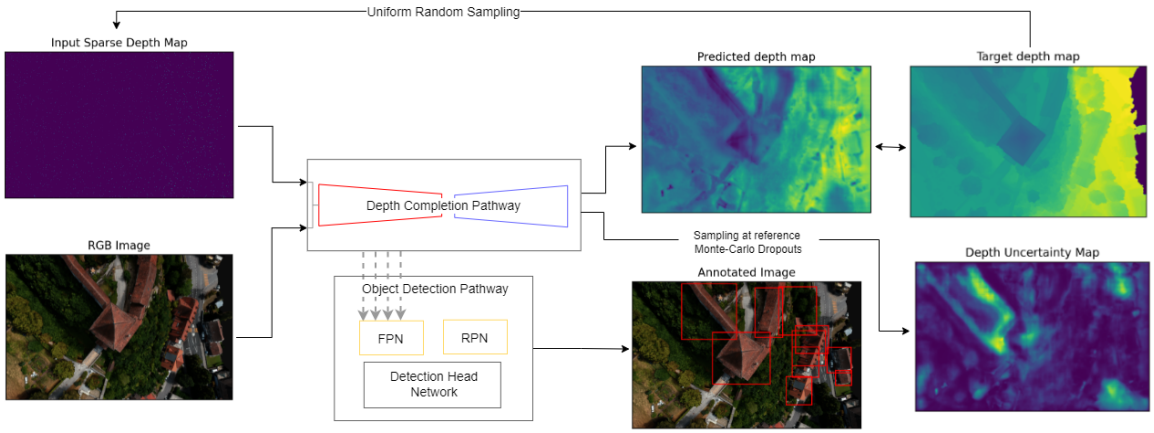

Figure 3.1: Overview of the proposed multi-task learning framework and the training process. Two tasks of depth completion and object detection are jointly carried out in an encoder-focused multi-task network in a hard parameter sharing fashion to introduce resource efficiency and more robustness to both tasks.

tackle this problem by processing sparse depth maps to estimate the missing depth values and output denser depth maps. These methods either use only the depth map (unguided depth completion), or benefit from auxiliary information provided by the RGB images acquired through a camera and a registered depth sensor (RGB-guided depth completion).

Depth inpainting is one of the early attempts to solve the depth comlpetion problem [60]. Although similar methods have proven useful for certain purposes, they lack necessary levels of generalizability and scalability required for outdoor inspections. Consequently, owing to their impressive performance, deep learning-based methods have become the main point of interest in the field. The state-of-the-art works present the use of the conventional and modified versions of Convolutional Neural Networks (CNNs) for depth completion [61, 62]. In both traditional and deep learning-based RGB image-guided methods, the core idea is to use the changes in the local geometric features, texture, and color of the scene to place the missing depth values and even correct the ones already existing in the map. The basic difference in these methods is the type of operations carried out to extract the scene features that help the re-

gression process. Respectively, the choice of the right geometric features to extract from the RGB image to guide the depth completion process in a deep learning framework remains an open research challenge. This study addresses this challenge and reflects on the typical objectives of a UAV inspection flight by proposing a multi-task network that achieves both depth completion and object detection in a single pass. The impact of incorporating object detection as an auxiliary task on the results of depth completion, as opposed to accomplishing the task using a single-task network, is explored. Object detection is chosen to represent a widely used application of UAV inspection flights and an inexpensive alternative to pixel-wise annotation and processing required for semantic segmentation, which is usually impractical for aerial data acquisition. Additionally, to study the change in the behavior of the network with employing the object detection pathway, the Monte-Carlo dropout technique [63] is used to generate pixel-wise uncertainty maps for both cases.

3.2 Background and Related Work

In this section, the necessary prerequisites as well as previous research in the context of this chapter are presented.

3.2.1 Depth Completion

Pixel-wise dense depth maps are necessary for a variety of applications, including virtual reality, film industry, autonomous driving, and 3D modelling [64]. Depth sensors such as LiDAR or RGBD cameras have been more reliable than monocular single image depth estimation methods for generating depth values of a scene, however, they are unable to produce high-density depth maps. To address the issue with the depth values that these sensors produce, many methods have been proposed en-

hancement [65], denoising and refinement [66], inpainting [60], and upsampling [67] of depth maps. These methods often involve depth maps that are of densities more than 20% up to 80% and usually deal with partial defects such as holes or noise. In what is called depth completion, depth maps of extremely low density (below 5%) are processed to produce dense depth maps. This process would face challenges when trying to estimate the missing depth values without a sense about the presence of the same object along a surface. As a result, guidance for this context would be provided from an auxiliary source which could be color from RGB images, semantic maps, constraints from surface normals, or object boundaries. Such methods are called guided methods and can be based on classical image processing or deep neural networks for extracting features or kernels that would help navigate from known depth values to missing ones [68]. In contrast, a subcategory of depth completion methods, known as unguided depth completion methods, do not benefit from auxiliary context guiding the process [69]. Despite improvements in the methodologies based on different variants of Convolutional Neural Networks (CNNs) [70] or embedded confidence maps [71], this field has not received as much attention as the guided depth completion methods due to the resultant uncertainty and unclear object boundaries.

3.2.2 Deep Learning-based Depth Completion Methods

Previous works that use RGB guidance for depth completion often aim to develop a network that learns explicit geometric features of the scene to help improve the quality of the completed depth map. These features can be either 2D or 3D representations of the scene which capture specific characteristics of the surface that help the network interpolate and predict more reliable depth values. Among such representations are surface normals that have received reasonable attention as they propose an added geometric constraint for the depth completion network [72, 73, 74]. [75] explored

different depth representations that contribute to depth completion and found the normals along surface boundaries to be most effective. In [76], a novel encoder-decoder-like model based on the well-known ResNet architecture [77] was deployed to directly derive a dense depth map from the input RGB image and sparse depth map. Due to the efficiency and comprehensibility of this architecture, it was found to be most suitable for integration with a second stream, i.e. the object detection stream, for the purposes of this study.

3.2.3 Object Detection

Object detection is a fundamental vision task defined as the task of localizing objects present in the scene and recognizing the associated classes [78]. Early works in the field used classic feature extractors which would lead to creating Histogram of Oriented Gradients (HOG) [79]. Although these models are the foundation of today's advanced object detection models, they lacked accuracy and generalizability and were not suitable for real-time applications. With rapid advancements in deep neural networks based on convolutional layers, the object detection task experienced improvements in terms of time efficiency, robustness, and accuracy. Datasets such as PASCAL Visual Object Classes (VOC) [80], Microsoft COCO [81], and Open Image [82] provided the developers with millions of images and object instances to challenge their networks in multiple criteria. These criteria include performance measurements in terms of precision, recall, frames per second (FPS), and mean Average Precision (mAP).

Object detectors in general are classified as single-stage traditional and deep learning-based detectors where the hand crafted feature extraction of traditional methods was replaced by the robust features derived by CNNs. Deep learning-based methods are categorized as two-stage or single-stage detectors which carry out the detection process in a "coarse-to-fine" fashion or complete it in one step [83]. A family

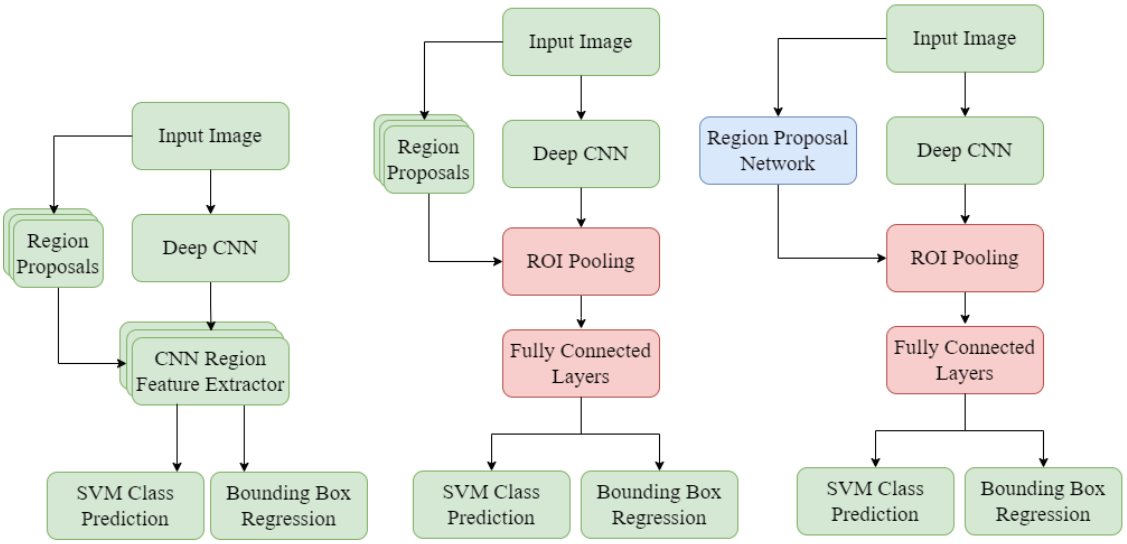

Figure 3.2: Architectures of the R-CNN family as examples of two-stage object detectors.

of models belonging to the two-stage detectors are the Region-based Convolutional Neural Networks (RCNN). First work belonging to this family started by generating object proposals using selective search [84]. These proposals are then resized to a fixed size and passed through a CNN pretrained on ImageNet. The CNN extracts features from each proposal. Then, linear SVM classifiers are employed to predict object presence and recognize categories. However, the high number of proposed regions and the overlaps between them would make this model time-inefficient. This led to the introduction of Spatial Pyramid Pooling Networks (SPPNet) whose contribution was generating fixed-length feature maps regardless of the size of the image or the proposed region at once [85]. Fast-RCNN [86] further improved the speed of detection by enabling detector and bounding box regressor at the same time in the same network. Later, Faster R-CNN [87] introduced an integrated Region Proposal Network (RPN) that generates region proposals directly from feature maps. This design allows Faster R-CNN to achieve improved speed and end-to-end training capabilities.

Figure 3.2 demonstrates simplified architectures of the RCNN family.

3.2.4 Multi-task Learning

Multi-task Learning (MTL) is an approach that aims to enhance generalization performance by leveraging domain-specific information from related tasks. It achieves this by training multiple tasks simultaneously while utilizing a shared representation. The shared representation enables the model to capture common patterns and transfer knowledge between tasks, leading to improved generalization [88]. In MTL, a single model is trained to learn to execute multiple tasks each with their own objective. This training leads the model to capture underlying patterns across the tasks while adapting to the unique characteristics of each task. In deep learning, MTL is usually carried out in a hard or soft parameter sharing manner [89].

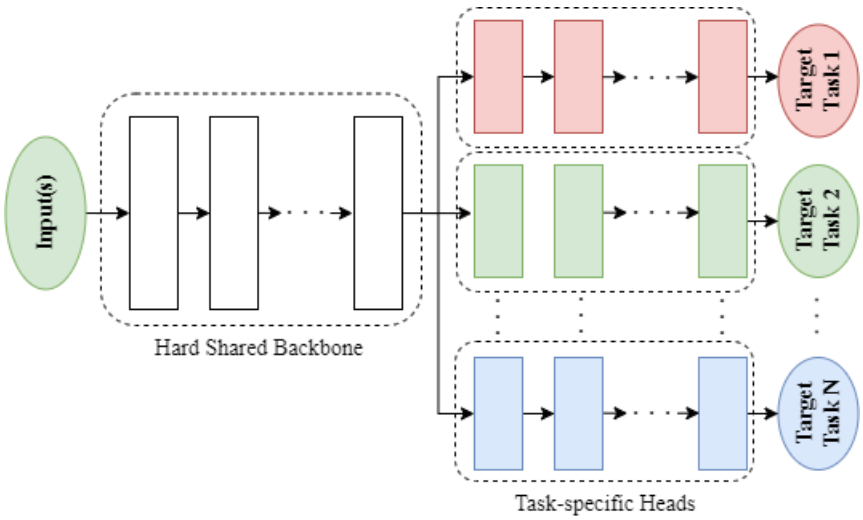

Figure 3.3: Hard parameter-sharing in deep neural networks.

In **hard parameter-sharing**, the hidden layers between between all the tasks are shared and each task has its own head network to learn the task-specific features and output the desired signals [88]. Figure 3.3 shows this parameter sharing technique in a custom neural network.

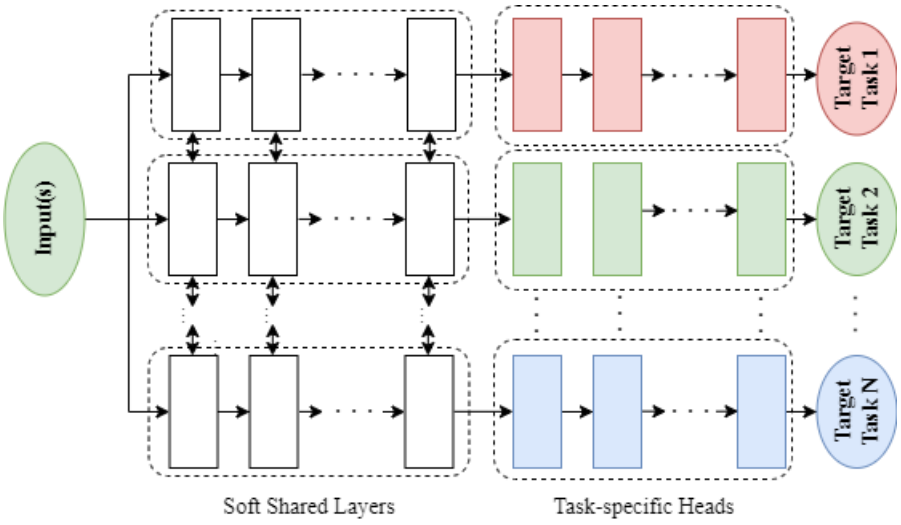

Figure 3.4: Soft parameter-sharing in deep neural networks.

In **soft parameter-sharing**, there is a model assigned to each task to learn the task, while the distance between the parameters of these models is regularized to promote similarity among them [90]. A representation of this method is depicted in Figure 3.4.

3.2.5 Multi-task Networks for Depth Completion

Multi-task learning is a promising machine learning area which takes advantage of auxiliary tasks for resource efficiency and reducing computation cost at inference time. Depending on the relevancy of the target tasks handled by a single network or different networks merged together, the performance quality of the network can experience an improvement or a decline compared to the case of using separate single-task networks. In this regard, [91] studied the efficacy of grouping different sets of tasks in computer vision to solve through multi-task learning. By guiding the network's loss function to suit a main task and an auxiliary task, they demonstrated improved results for the main task, which could come at the cost of degraded results for the auxiliary task. Although the use of multi-task networks prevents re-learning certain features

for closely-related tasks and proves its efficiency in the use of resources, it comes with the trade-off between the model performance and resource gain [92]. While choosing the tasks to be included in the network, it is also necessary to consider the requirements for the corresponding application. In recent years, multi-task learning together with multi-sensor fusion have been used in 2D and 3D perception systems to improve the computation cost at inference time and the system performance. In [93], a joint model was proposed which successfully combined four tasks, with 3D object detection as the target task to be improved, which demonstrated the complementary natures of mapping, object detection, and depth completion. They trained a multi-task multi-sensor detector in an end-to-end fashion which achieved dense point-wise and ROI-wise feature fusion. [94] improved both tasks of semantic segmentation and depth completion by using a feature-sharing encoder and a three-branch decoder.

3.3 Methodology

Most of the work in traditional and deep learning-based depth completion focuses on scene representations containing features that are local to the individual pixels and ignore the neighborhood of the pixel that make an entity of a semantic object. In such cases, texture and color information would be utilized to account for the presence of the same object along a surface [60], or the network would work to extract specific geometric features such as object boundaries and edges [95] or surface normals [72]. Such strategy requires added computations and supervision to lead the network to learn such representations. This, in turn, hints that the extracted features would always lack a relevance to the certain spatial relationships existing in similar entities of semantic objects belonging to the same category. In a more practical context considering a UAV inspection flight, if the network identifies an object, e.g. a bridge,

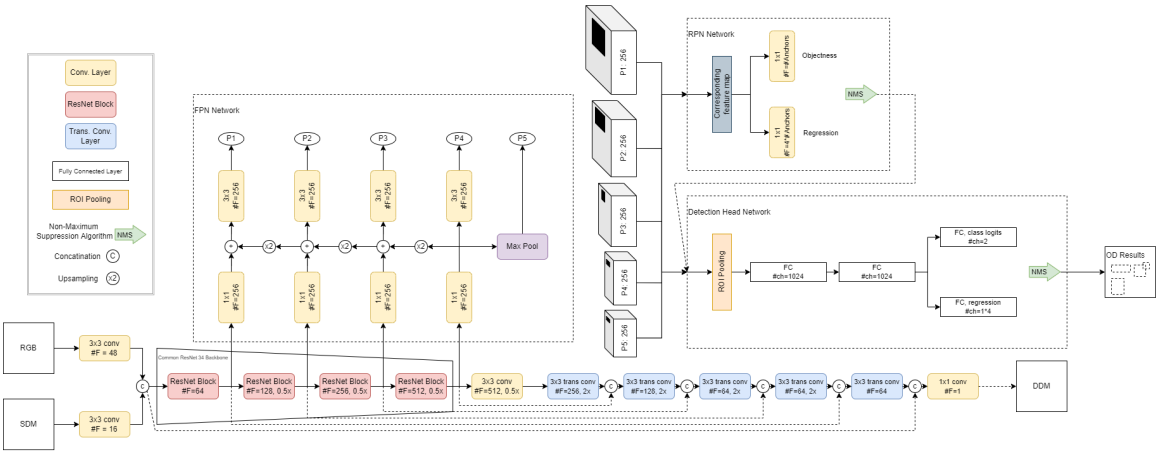

Figure 3.5: The proposed architecture for the multi-task network. Features from different stages of encoding the fused data from the RGB image and the depth map are shared between the object detection pathway and the depth completion pathway.

the network can use its contextual knowledge of the bridge's probable shape and size to estimate the expected depth values for pixels along the sides of it. This will also pave the way for the network to reach a more complete 3D reconstruction of the scene in case occlusion poses a challenge. With this as the core idea of the presented study, a proposal is made to guide the extracted features along the depth completion stream to satisfy a second task, namely object detection, through a multi-level shared backbone. Figure 3.1 represents an overview of the proposed model.

The work is executed by employing separate sub-networks, referred to as pathways or streams, in a multi-task model where extracted features from several stages of encoding throughout the backbone are fed to two task-specific heads. While the depth completion network tends to extract features more local to the pixels and of geometric value, the object detection stream extracts features that are more unique to the objects in the scene with higher semantic value. Thus, the parameter sharing allows the backbone to learn a representation space that benefits both tasks, owing to the added supervision. As well, in order to further study the behavior of the multi-task network compared to its single-task opponent, a representation for the

uncertainty level of the depth maps generated by each of the networks is required. This is obtained by using Monte-Carlo dropouts inspired by [63]. The following sections elaborate on the separate pathways, the feature sharing method for multi-task learning, and the corresponding pixel-wise uncertainty estimation. The details of the model architecture are shown in Figure 4.3.1.

3.3.1 Depth Completion Pathway

The core architecture for the depth completion pathway is inspired by [76] and composed of an encoder-decoder structure. First, the sparse depth map and the RGB image are fed to two different convolutional layers with filters of 16 and 48 channels. The resultant feature maps are concatenated and the merged features go through five different stages of down-sampling using residual blocks of ResNet-34 [77] and a convolutional layer with 512 channels. The encoded features are then up-sampled using five transpose-convolutional layers of 256, 128, 64, 64, and 64 channels, respectively. The outputs of each of the down-sampling stages from the encoder are concatenated with the resultant features of the corresponding up-sampling stages in the decoder and are fed to the next stage. The output is a dense depth map image of the same size as the inputs.

3.3.2 Object Detection Pathway

The object detection pathway is designed to serve as the auxiliary stream in this multi-task learning scheme. Inspired by [96], the pathway uses a ResNet-based Faster R-CNN detector augmented by a Feature Pyramid Network (FPN) neck. Intermediate feature maps from the shared backbone are fed to the pathway on 4 different levels. The FPN pulls out the ResNet blocks' outputs and provides 5 different scale feature maps for the next modules. The FPN features are then fed into a Region

Proposal Network (RPN) where objectness probability and box regression proposals are produced. These feature maps from the FPN are then used to predict bounding boxes corresponding to the different scale they are generated with. The different anchor boxes from corresponding feature maps are fed to the RPN to provide the proposals. The final detection head then refines the proposed bounding boxes through a Region of Interest (ROI) pooling layer. The outputs of the RPN and the final detection head are then fed to a Non-maximum Suppression (NMS) algorithm to remove the duplicate bounding boxes.

3.3.3 Model Uncertainty Representation

To facilitate a comparative study of the behaviors exhibited by the single- and multi-task networks, a key objective is to quantify the uncertainty present in the model's outputs. This endeavor is inspired by the seminal work introduced in [97] where a Bayesian approximation for model uncertainty was obtained using active dropouts at inference time. The same method is used to obtain the predictive mean and variance for the model output by fitting a Gaussian distribution to the results of multiple forward passes. Subsequently, for each pixel in the map, a corresponding variance value in the uncertainty map is visualized, as shown in Figure 3.10, as well as a mean to represent the inferred depth result.

3.3.4 Learning Objectives

Depth Completion Pathway Learning Objectives

The learning objectives for this network target the quality of the generated dense depth map in different aspects, for each of which a loss function is defined.

Depth Consistency: A dense depth map provides direct supervision for the network

by acting as the ground truth. In this work, L2-norm, or the Root Mean Squared Error (RMSE), is used to keep track of the consistency of the completed depth given by Equation 3.1:

$$l_{consistency} = \frac{1}{n}\sum_{i=1}^{n} ||\hat{Y}_i - Y_i||_2 \tag{3.1}$$

where n is the number of pixels and Y_i and $\hat{Y}_i$ represent predicted and ground truth depth values of the i^{th} pixel, respectively.

Depth Smoothness: To guarantee better depth completion quality, it is important to reduce the noise in the depth map by taking into account the gradient of the depth values in the x- and y-axis on the plane of the depth map. That is implemented by taking the L1-norm of the second order derivative of the depth values. This ensures that any irregularities in the depth map are suppressed (Equation 3.2).

$$l_{smoothness} = \frac{1}{n}\sum_{i=1}^{n} |\partial_x^2 \hat{Y}_i| + |\partial_y^2 \hat{Y}_i| \tag{3.2}$$

Object Detection Pathway Learning Objectives

Based on the original Faster R-CNN loss [87] and its implementation in TorchVision [98] detection framework, a detection loss is utilized which contains two terms: proposal loss and final detection loss which are summed with a relative weight λ (Equation 3.3).

$$l_{detection} = l_{proposal} + \lambda l_{final\ detection} \tag{3.3}$$

This loss is a combination of the RPN loss ($l_{proposal}$) and the Fast-RCNN loss ($l_{final\ detection}$). The RPN is responsible for generating valid region proposals with bounding boxes around potential objects. It is trained to accurately predict an objectness score and the bounding box regression offsets. Therefore, its loss is composed of two component; a classification loss which is simply a binary cross-entropy loss for

identifying a background or object label (Equation 3.4), and a regression loss which is an L1-norm for predicting bounding box offsets (Equation 3.5).

$$l_{\text{RPN\_cls}} = -\frac{1}{N_{\text{cls}}} \sum_i \sum_j \left(y_{ij} \log(p_{ij}) + (1 - y_{ij}) \log(1 - p_{ij})\right) \tag{3.4}$$

$$l_{\text{RPN\_reg}} = \frac{1}{N_{\text{reg}}} \sum_i \sum_j \sum_k \text{smooth}_{L1}(t_{ijk} - t^*_{ijk}) \tag{3.5}$$

And the total loss for RPN will be a combination of the two components as $l_{proposal} = l_{\text{RPN\_cls}} + l_{\text{RPN\_reg}}$ [87]. The Fast-RCNN loss combines the classification loss accounting for the difference between predicted class probabilities and ground truth labels using softmax cross-entropy (Equation 3.6) and the regression loss for measuring the difference between bounding box regression offsets and ground truth boxes using L1 loss (Equation 3.7) [86].

$$l_{\text{cls}} = -\frac{1}{N_{\text{cls}}} \sum_i \sum_j \left(y_{ij} \log(p_{ij}) + (1 - y_{ij}) \log(1 - p_{ij})\right) \tag{3.6}$$

$$l_{\text{reg}} = \frac{1}{N_{\text{reg}}} \sum_i \sum_j \text{smooth}_{L1}(t_{ij} - t^*_{ij}) \tag{3.7}$$

The Fast-RCNN loss is also a linear sum of its two components as $l_{finaldetection} = l_{\text{cls}} + l_{\text{reg}}$. The proposed network is trained with a weighted combination of the explained losses for the two streams. In the above equations, variables y_{ij}, p_{ij}, t_{ijk}, t^*_{ijk}, t_{ij}, and t^*_{ij} represent the ground truth labels, predicted probabilities, ground truth box offsets, predicted box offsets, for the respective losses. N_{cls} and N_{reg} present the normalization factors for classification and regression losses.

3.4 Experiments and Results

3.4.1 Implementation Details

The proposed network including its two pathways is implemented in PyTorch [99] using the Lightning modules [100]. Adam is used as the optimizer with a starting learning rate of 10^{-4} which is multiplied by 0.5 after every 5 epochs. The networks were trained for 20 epochs using a NVIDIA GTX 3060 with up to 16 GB of memory. The input images (RGB and depth map) were fed to the network after being resized to 320×240 in separate batches of size 1. The sparse depth maps were generated by uniformly sampling 0.7% of points from the dense ground truth depth maps.

3.4.2 Dataset

The Aerial Depth Dataset developed by [101] was utilized, consisting of RGB and depth images of infrastructure models generated using a photogrammetry software. Due to the lack of available annotations for performing the object detection task on this dataset, a manual annotation process was conducted to annotate the buildings and bridges in 1000 frames.

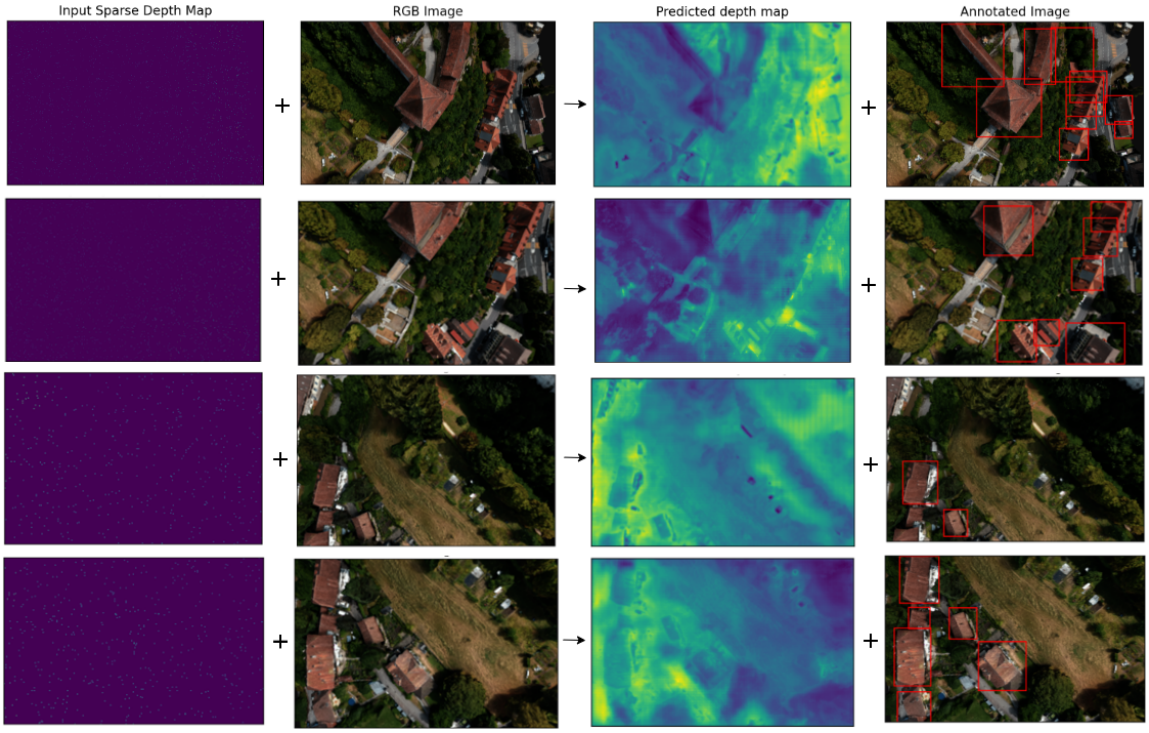

Figure 3.6: Depth completion and object detection results for the proposed multi-task network.

3.4.3 Comparative Analysis

To test the developed hypothesis mentioned in Section 3.3, the single-task and proposed multi-task networks were trained and tested using identical hyper-parameters to compare the quality of the output depth maps and assess their robustness in response to defective inputs. Figure 3.6 demonstrates the input-output summary of the multi-task network and the results are compared to those of the single-task network as depicted in Figure 3.7. From a qualitative perspective, it can be seen that the output depth map from the multi-task network has done a better job at replacing depth values around the objects in the scenes that have been annotated in the ground truth images, which is apparent from clear object boundaries in the generated depth maps.

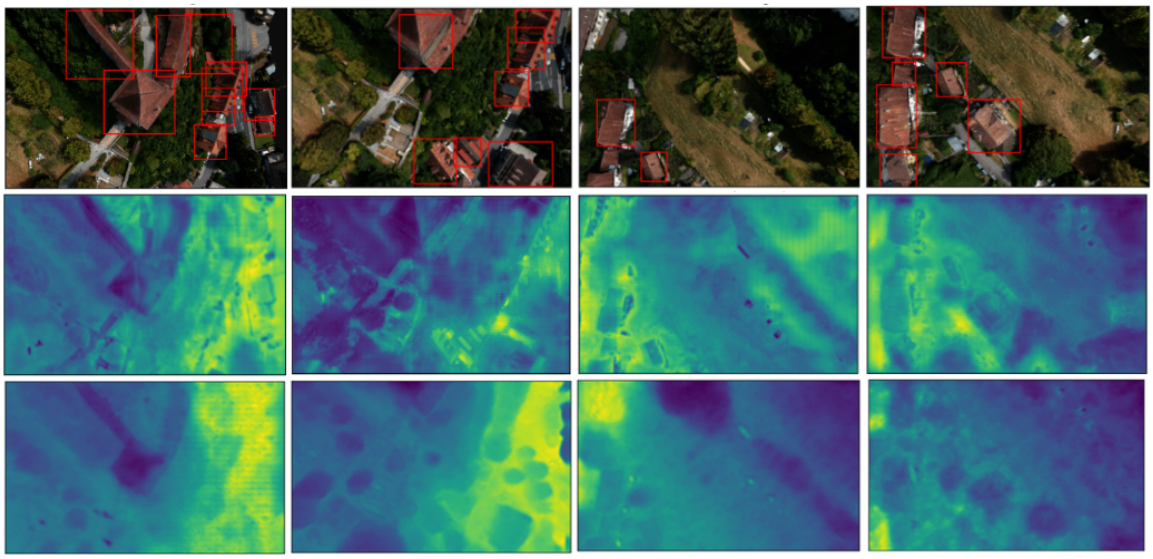

Figure 3.7: Comparison between the output depth maps of the single- and the proposed multi-task networks. Notice the difference in the object boundaries. The multi-task network (second row) identifies comparatively crisp boundaries in areas where an object has been detected. The single-task network (third row), in contrast, predicts blurred depths.

3.4.4 Robustness Analysis

As discussed before, the added supervision guides the shared backbone layers to extract contextual information from the inputs, ultimately leading to enhanced performance in depth completion. To investigate how this context could improve the robustness of the proposed network, the single- and multi-task networks were subjected to defective input depth maps for evaluation and analysis. Firstly, the networks were exposed to RGB images with sparse depth maps that contained increased Gaussian distance-dependent noise. The results of this experiment are depicted in Figure 3.8. Then, random boxes are introduced on the maps which result in missing depth values within the boxes to test if the relative depth of the structures can be inferred even in the absence of their actual depth values. Figure 3.9 shows two samples of the results of this experiment. From a qualitative standpoint, it is evident that the multi-task network is more effective at predicting missing depth values, particularly

those associated with structures, compared to the other network.

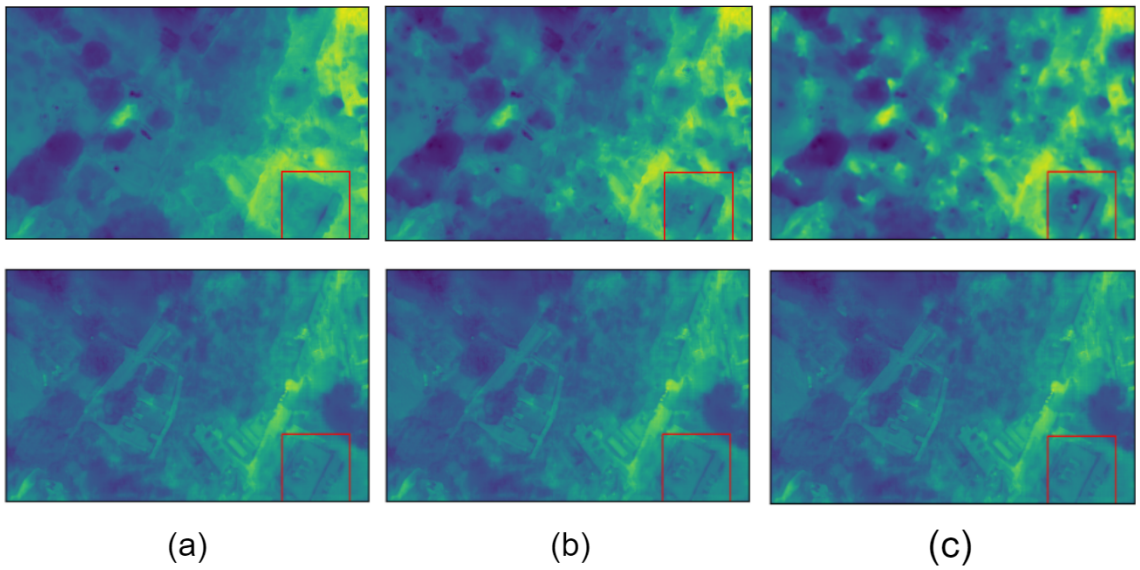

Figure 3.8: Comparing the noise robustness of the single- and multi-task networks. Gaussian distance-dependent noise is applied to the input sparse depth maps with increasing variance from (a) 10% to (b) 20% and (c) 40% of the actual values. First and second row show the results of the single-task and the multi-task networks, respectively.

3.4.5 Uncertainty Map Analysis

Figure 3.10 demonstrates the low variance in uncertainty outputs of the multi-task network when compared to the single-task counterpart. On visual inspection, the uncertainty maps of areas predicted between the bounding boxes, e.g., buildings, present significantly less uncertainty. The uncertainty estimation through the proposed network for unannotated areas e.g., trees, remains higher than the annotated areas. In contrast, with the single-task network, objects like buildings have higher uncertainty which is detrimental to UAV inspection missions.

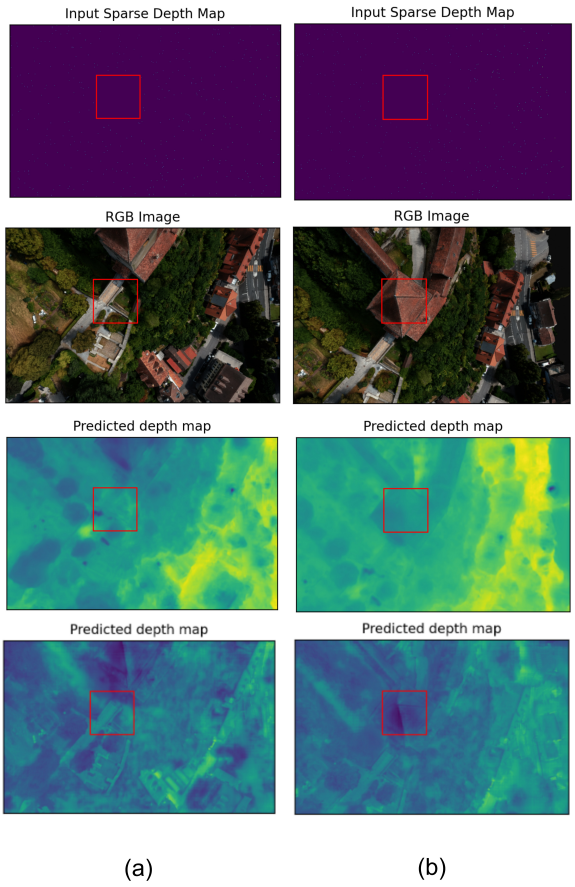

Figure 3.9: Comparing the performance of the two networks in response to incomplete sparse maps as inputs with missing values in the marked box for two different (a) and (b) scenes. The third row corresponds to the outputs of the single-task network and the fourth row demonstrates the outputs of the multi-task network.

3.5 Conclusions and Future Work

To address the issue of sensor data defects that can impact both the 3D modelling and safety insurance of autonomous aerial vehicles, a multi-task network to attain the tasks of depth completion and object detection was proposed and implemented. Semantic feature maps were introduced as representations of the scene, leveraging their potential to enhance the results of depth completion. The network runs on a shared backbone with two task-specific heads assigned to respectively produce a dense depth map and bounding boxes localizing the infrastructure present in the scene. A

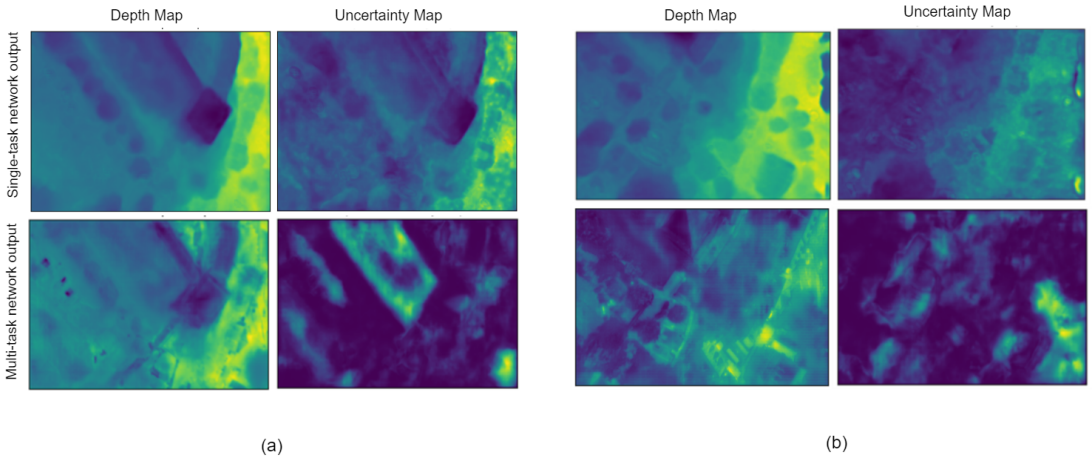

Figure 3.10: Uncertainty map comparison between the depth maps generated by the single- and proposed multi-task network for two different (a) and (b) scenes. The first columns are the output depth maps and the second columns are the uncertainty maps for the singe- and multi- task network results in the first and second rows.

comparative analysis was carried out on the results of the single- and the proposed multi-task networks which proved enhanced performance of the multi-task network in producing depth maps for the pixels belonging to the detected objects by the object detection stream.

Chapter 4

How to manipulate the path to see better: A Deep Reinforcement Learning Approach to Appearance-based View Planning for 3D Reconstruction

4.1 Introduction

Active vision is characterized by the ability of a robot to make decisions about placing or reconfiguring its sensors to complement its perception of the environment [102]. This ability leads to meaningful actions of the robot based on interpretations of its surrounding environment that it has proposed so far via its previous sensor outputs. Active vision grants the robot a planning strategy, namely view planning, for actively placing its sensors in different viewpoints to uncover most amount of information about the target. In the context of active 3D reconstruction, view planning is used to optimize the robot's path until the task requirements are satisfied. For the application of 3D reconstruction of infrastructure using UAV-based imaging which is the concern of this work, the view planning problem dictates the data acquisition process and significantly impacts the reconstruction results. Previous work in this domain either relies on a given proxy of the target to build upon while planning the views [103, 104] or generates a partial reconstruction using the knowledge of the agent about the

target so the camera is navigated to complete the model [105, 106]. In this setting, the agent iteratively calculates the next waypoint to attend where it can capture the next-best-view (NBV) with the highest predicted information gain. However, when the purpose is to capture views from newly recognized targets or in the case of targeting complex structures, a geometric proxy of the target might not be accessible and online 3D reconstruction to achieve guidance can be computationally expensive and time-inefficient. On the other hand, under the assumption that adequate computation resources exist onboard the drone, algorithms that use an external model for guidance purposes mostly focus on the coverage completeness of the area where less attention is paid to the relative visual information contained in consecutively captured views.

In this chapter, a novel approach to fully appearance-based view planning for online NBV planning is proposed. The key feature of this work lies on its model-free nature that makes it independent of the true state of the environment, namely the actual 3D model, both during training and inference time. This allows the method to be applied to a wide range of settings and customized to fit different applications. The concept is introduced by drawing parallels between the computational representation of views and how humans perceive objects. This is achieved by highlighting the process of recognizing and interpreting distinct visual cues that enable humans to perceive and understand objects [107]. Inspired by this concept, a Soft Actor-Critic (SAC) method [108] is utilized to train an agent with the objective of capturing the target from views that induce a more substantial change in its perception of the environment thus far. The core idea of this method is to use the local visual features of the scene and their positioning to guide the agent to predict the NBV that would result in revealing the highest number of unseen visual features as the agent remembers them. Despite the prominence of trajectory optimization and view planning in various fields, including 3D reconstruction and condition assessment, the

specific focus on optimizing trajectories solely based on appearance cues has not been visited to the fullest. The scarcity of prior research investigating this intersection of appearance-based planning and trajectory optimization posed a significant challenge in benchmarking and comparing the proposed approach against established methods. In light of this, this work uses a baseline comparison to showcase the efficacy of the proposed method. This baseline includes reconstruction results from full scans of the target in the simulation environment. Subsequently, the proposed model aims at handling the trade-off between the trajectory length of the image capturing process and the reconstruction results from the captured views.

4.2 Background and Related Work

4.2.1 Markov Decision Process (MDP)

A Markov Decision Process (MDP) is built upon a Markov Process and a Markov Reward Process [109]. A Markov Process (MP) is a discrete-time stochastic process describing a sequence of probable states $\{S_1, S_2, ..., S_t\}$ where each state only depends on the last state and not the states before that, or:

$$p(S_{t+1}|S_t) = p(S_{t+1}|S_1, S_2, ..., S_t) \tag{4.1}$$

This process is stochastic, meaning the transition between the states follows a probability $p(s'|s) = p(S_{t+1} = s'|S_t = s)$. Thus, in order to fully describe a Markov Process, a tuple $< S, P >$ should be known where S and P represent the set of states and the transition probability between the states, respectively.

The Markov Reward Process (MRP) is defined by adding discounted reward to

the MDP and creating a tuple of $< S, P, R, \gamma >$ where:

$$r(s', s, a) = \mathbb{E}[R_{t+1}|S_t = s] \tag{4.2}$$

At last, the MDP is defined by adding the actions A to the last tuple to create $< S, A, P, R, \gamma >$ and formulate the reward as below:

$$r(s', s, a) = \mathbb{E}[R_{t+1}|S_t = s, A_t = a] \tag{4.3}$$

Stochastic policy $\pi(a|s)$ is the probability of choosing action a while observing state s where $\forall a \in A$ and $\forall s \in S$. Finding the optimal policy π^* that leads to maximum reward over the course of an episode is the main goal of solving an MDP.

MDP is the classic framework for describing a Reinforcement Learning problem. According to this framework, actions of an agent influence the immediate reward associated with that action at that state and the preceding states and rewards. The goal is that the agent learns to choose the optimal action to perform based on the part of the whole state of the environment that it has access to through its observations. This framework makes it possible to model any sequential decision-making problem in a manner that enables the agent to learn how to act under uncertainty through experience [110]. Figure 4.1 depicts the main components of a Markov Decision Process in a typical reinforcement learning cycle.

4.2.2 Discounted Expected Reward

Taking the actions that maximize the immediate reward will not result in long-term success of the agent in reaching the goal. Instead, the agent must take actions that maximize the sum of the rewards through all the steps to achieving the goal. The discounted expected return is the sum of rewards obtained at each time step, weighted

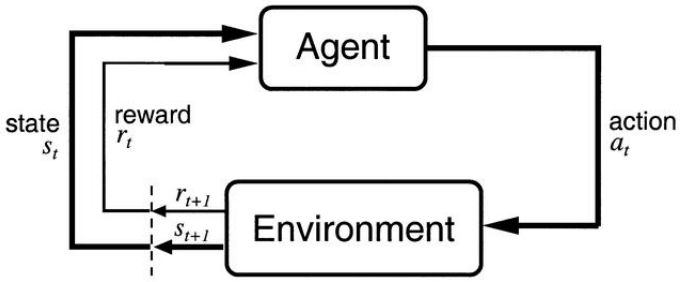

Figure 4.1: Main components of a Markov Decision Process in a typical reinforcement learning cycle adopted from [2].

by the discount factor raised to the power of the time step and is calculated as below.

$$G_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1} \tag{4.4}$$

Here, γ is the discount factor and is applied to assign lower priority to the reward of future step compared to the immediate rewards. A discount factor of 0 would mean that only immediate rewards are valued, while a discount factor of 1 would treat all future rewards equally. The range of the above summation is in the range $(0, T)$, indicating that the sum extends from the current time step (t=0) indefinitely into the future. The task for which the MDP is being formulated can be either *episodic*, meaning the process ends as the agent reaches the goal ($T = N$ with N being the number of steps to reach the terminal state), or *continuous*, meaning the agent interacts with the environment in a continuous time domain and there are no explicit episode boundaries ($T = \infty$).

4.2.3 Value Functions

The value function assesses the desirability or utility of being in a particular state or taking a specific action in the MDP. There are two main types of value functions: the state value function (v) and the action value function (q).

State Value Function is a measure of the expected cumulative discounted reward that the agent receives in case it starts from state s and follows the policy π thereafter.

$$
\begin{aligned}
V_\pi(s) &= \mathbb{E}_\pi \Big[G_t | S_t = s \Big] \\
&= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right] \\
&= \mathbb{E}_\pi \left[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} | S_t = s \right] \\
&= \underbrace{\sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)}_{\text{Sum of all probabilities } \forall \text{ possible } r} \\
&\quad \left[r + \gamma \underbrace{\mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+2} | S_{t+1} = s' \right]}_{\text{Expected reward from } s_{t+1}} \right] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \Big[r + \gamma V_\pi(s') \Big]
\end{aligned}
\tag{4.5}
$$

The state value function gives insight about the overall utility of different state.

Action Value Function determines the long-term utility of taking action a in state s as it measure the expected cumulative discounted reward that the agent receives in case it starts from state s and takes action a and follows the policy thereafter.

$$
\begin{aligned}
q_\pi(s,a) &= \mathbb{E}_\pi\left[G_t|S_t=s,A_t=a\right] \\
&= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t=s,A_t=a\right] \\
&= \mathbb{E}_\pi\left[R_{t+1}+\gamma\sum_{k=0}^{\infty}\gamma^k R_{t+k+2}|S_t=s,A_t=a\right] \\
&= \sum_{s',r} p(s',r|s,a)\left[r+\gamma\mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+2}|S_{t+1}=s'\right]\right] \\
&= \sum_{s',r} p(s',r|s,a)\left[r+\gamma V_\pi(s')\right]
\end{aligned}
\tag{4.6}
$$

The action value function determines the quality of taking an action in a certain state.

4.2.4 Reinforcement Learning

Machine Learning (ML) is an area of Artificial Intelligence (AI) in which the main concern is creating learners that can acquire intelligence through data. How the data is acquired and how the learner utilizes the data in order to learn the specific task in hand are what make the three main sub-fields of Machine Learning; Supervised Learning (SL), Unsupervised Learning (UL), and Reinforcement Learning (RL) [110]. The methodology of the rest of the thesis is mainly concerned with Reinforcement Learning. Reinforcement Learning is a topic in machine learning according to which the learner is trained based on the consequences of its actions. The actions of the learner therefore affect the reward that it receives, the state of the learner in the environment, possibly the state of the environment, and thus the future rewards it will receive as results of its actions. This implies the two essential characteristics of

this type of learning; learning through trial-and-error and delayed reward [2]. On the other hand, in Supervised Learning, the act of learning is from previously acquired and labeled data which leaves room for the learner to choose its own actions and learn from them. The main purpose in this type of learning is to generalize the learner's intelligence to respond to situations that the learner has not experienced in the training phase. Finally, in Supervised Learning the learner is exposed to already acquired unlabelled data to find certain behaviour or structure in the data. This might be useful for already gathered data but does not take into account a reward signal that would train the learner to respond to its state [2]. This, again, takes away from the agent an opportunity to decide on its action which acts as the training data as it accumulates.

In Reinforcement Learning, the goal is to find an optimal policy π^* that leads to higher value functions resulting from any other policy π for $\forall s \in S$. The optimal action value function q^* will then become:

$$q^*(s,a) = \max_{\pi} q^{\pi}(s,a) \tag{4.7}$$

which can give the optimal state value function v^*

$$v^*(s) = \max_{a \in A(s)} q_{\pi*}(s,a) \tag{4.8}$$

4.2.5 Monte Carlo Methods

By simulating episodes and averaging the obtained rewards, Monte Carlo (MC) methods use the experience from the episodes to learn. Two known types of Monte Carlo predictions are the *first-visit MC method* and the *every-visit MC method* [109]. While the former averages the returns of the first visit to each state, the latter averages the

returns from all visits to each state.

More important application of Monte Carlo methods is to approximate optimal policies. Beginning with an arbitrary policy, complete evaluations of the policy are followed by policy improvement steps until the optimal policy is reached. For each step k, the greedy policy $\pi(s) = \text{argmax}_a\, q(s,a)$ chooses the best action that it perceives by considering the action-values involving the current state s. Then, π_{k+1} is constructed by taking into account q_{π_k} for each state. This ensures that the new policy is always the same as or better than the last policy.

4.2.6 Temporal Difference Methods

Temporal Difference (TD) methods combine the advantages of MC methods and dynamic programming by incrementally updating value functions after each time step using bootstrapping and intermediate estimates. The update rule for the state value function for TD methods is as below:

$$v(s_t) \leftarrow v(s_t) + \alpha\left(R_{t+1} + \gamma v(s_{t+1}) - v(s_t)\right) \tag{4.9}$$

where $v(s_{t+1})$ is the estimate for the value of the next state and α is the learning rate determining the update step size.

Q-Learning is one of the popular TD methods that estimates the action value function $Q(s,a)$. It updates the Q-value with the maximum Q-value of the next state with the update rule below:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left(R_{t+1} + \gamma \max_{a'} Q(s_{t+1},a') - Q(s_t,a_t)\right) \tag{4.10}$$

4.2.7 Deep Q-Learning

Problems that involve complex environments, defining a table to hold all the Q-values is not practical. This is where a Deep Neural Network (DNN) can work efficiently as function approximator with parameters θ to estimate the Q-values $Q(s, a; \theta) \approx Q^*(s, a)$. To train such network, the TD error $y_i - Q(a, a, ; \theta_i)$ is used as below:

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(e)} \left[(\underbrace{y_i}_{\text{target}} - \underbrace{Q(s, a; \theta_i)}_{\text{prediction}})^2 \right] \tag{4.11}$$

where y_i is the TD target and is defined as:

$$y_i = r + \gamma \max_a Q(s', a'; \theta'_{i-1}) \tag{4.12}$$

Two important key concepts in the above equations are $U(e)$ and $Q(s, a; \theta')$. $U(e)$ is the **experience replay** which holds the transitions $< s, a, r, s' >$ in a replay buffer during data collections. While training, a mini-batch of these transitions are used for loss and gradient computations. This introduces resource efficiency and exposes the training agent to a variety of transitions other than the last one, which was the case for the standard Q-Learning. In addition, experience replay stabilizes the training and avoids the correlation between consecutive experiences.

Another successful concept introduced in Deep Q-Network (DQN) was the use of a *target Q-network* ($Q(s, a; \theta')$) in addition to the *online Q-network* ($Q(s, a; \theta)$). The two networks have the same architecture, but different parameters. The online Q-network is upated using the loss in Equation 4.11 while the target network is updated using the parameters of the online network every K steps. This helps avoid instability during the training process.

4.2.8 Policy Gradient Methods

Policy gradient methods directly update the policy using gradients instead of approximating value functions. These methods aim to maximize the probability of the actions that maximize the expected cumulative reward without referring to value functions and therefore directly reach the optimal policy. With a policy π_{θ_k} with parameters θ at the k^{th} training step, the parameters of the policy are updated based by stochastic gradient ascent:

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta \pi_{\theta_k} \tag{4.13}$$

where $J(\pi_\theta)$ is the expected undiscounted return for the current policy and its gradient is calculated as below:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) A^{\pi_\theta}(s_t, a_t) \right] \tag{4.14}$$

In this equation, $A^{\pi_\theta}(s_t, a_t)$ is the advantage function for this policy. Advantage function is calculated by taking the state value off the Q-value following the policy. It quantifies the advantage of an action relative to the average action in a given state.

Actor-Critic Methods

One of the most popular policy gradient algorithms is the Actor-Critic (AC) methods. As discussed before, the policy model and the value function are two critical components of policy gradient algorithms. In the AC method, the value function is learned in addition to the policy function. Such methods contain two main parts;**Actor** which learns the policy π_{theta} guided by the **Critic** which learns the value functions for actions or the states. The interactions between different components of models based on the AC method is presented in Figure 4.2. This method leverages the policy

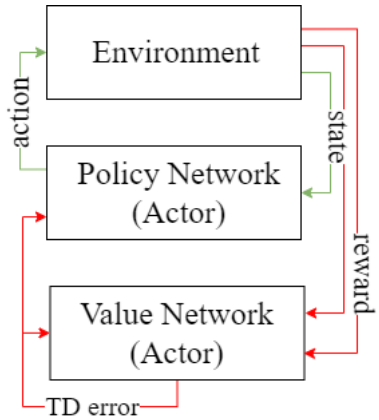

Figure 4.2: Training and testing cycle in actor-critic method adapted from [2]. The green and red arrows correspond to the inference and training phases, respectively.

gradient to update the actor's parameters. This gradient measures the gradient of the expected cumulative rewards with respect to the policy parameters. Values used to calculate this gradient use the output values of the critic which is updated using TD or MC estimation where the target values are computed using the observed rewards and estimated future values. This joint learning leads the actor to learn from the critic's feedback which the critic tends to learn more accurate estimates of the value function while observing actor's actions.

4.2.9 Soft Actor-Critic

The chosen algorithm for this problem is the Soft Actor-Critic (SAC), an off policy maximum entropy deep reinforcement learning approach with a stochastic actor [108] which works great for continuous action spaces where the environment is not or cannot be modelled. This algorithm concurrently learns the policy with two Q-functions. The main feature of SAC is entropy regularization, according to which the agent learns the policy by maximizing the trade-off between the expected return and entropy, i.e. the randomness in the learned policy. In reinforcement learning methods based on *entropy-regularization*, the agent is trained to tend towards policies leading to higher

entropy at each time step. This can be formulated as below:

$$\pi^* = \arg\max_{\pi} \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t\Big(R(s_t,a_t,s_{t+1}) + \alpha H\left(\pi(\cdot|s_t)\right)\Big)\right] \tag{4.15}$$

The exploration-exploitation trade-off is controlled by the entropy regularization coefficient, α, which can be fixed or entropy-constrained variant. With the above formulation, the state and action value functions will are calculated as below:

$$V^{\pi}(s) = \mathop{\mathbb{E}}_{a\sim\pi}[Q^{\pi}(s,a)] + \alpha H\left(\pi(\cdot|s)\right) \tag{4.16}$$

$$Q^{\pi}(s,a) = \mathop{\mathbb{E}}_{s'\sim P}[R(s,a,s') + \gamma V^{\pi}(s')] \tag{4.17}$$

Using the definition of entropy and replacing the expectation by its approximation which is the immediate sample, Equation 4.17 can be rewritten as below:

$$Q^{\pi}(s,a) \approx r + \gamma\left(Q^{\pi}(s',\tilde{a}') - \alpha\log\pi(\tilde{a}'|s')\right), \qquad \tilde{a}'\sim\pi(\cdot|s') \tag{4.18}$$

As a result, the loss function for the Q-function estimators can be obtained by the below equation.

$$L(\phi_i,\mathcal{D}) = \mathop{\mathrm{E}}_{(s,a,r,s',d)\sim\mathcal{D}}\left[\left(Q_{\phi_i}(s,a) - y(r,s',d)\right)^2\right] \tag{4.19}$$

where

$$y(r,s',d) = r + \gamma(1-d)\left(\min_{j=1,2} Q_{\phi_{\mathrm{targ},j}}(s',\tilde{a}') - \alpha\log\pi_{\theta}(\tilde{a}'|s')\right), \qquad \tilde{a}'\sim\pi_{\theta}(\cdot|s') \tag{4.20}$$

In this method, the Mean Squared Bellman Error (MSBE) loss is used to train the Q-functions. The MSBE loss measures the discrepancy between the Q-function's

predictions and the target values. The choice of which Q-function to use in computing the sample backup is crucial. SAC employs the clipped double-Q trick to address the overestimation bias issue in Q-learning. The clipped double-Q trick involves having two Q-function approximators: Q_{ϕ_1} and Q_{ϕ_2}. To compute the sample backup, SAC takes the minimum Q-value between Q_{ϕ_1} and Q_{ϕ_2}. This approach reduces the risk of overestimating the Q-values and improves the stability of the learning process.

To update the policy network during training, the actions are calculated using the reparameterization trick using the output mean and logarithm of standard deviation from the actor network.

$$\tilde{a}_\theta(s,\xi) = \tanh\left(\mu_\theta(s) + \sigma_\theta(s) \odot \xi\right), \qquad \xi \sim \mathcal{N}(0, I) \tag{4.21}$$

Then, minimum between the outputs of Q_{ϕ_1} and Q_{ϕ_2} as well as the entropy of the policy are used to update the policy network (Equation 4.22) [111, 112].

$$\max_\theta \mathop{\mathbb{E}}_{s\sim\mathcal{D},\xi\sim\mathcal{N}} \left[\min_{j=1,2} Q_{\phi_j}(s, \tilde{a}_\theta(s,\xi)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s,\xi)|s)\right] \tag{4.22}$$

4.2.10 Solutions to the View Planning Problem

Depending on the amount of information about the environment, such as the availability of a reliable model, and computation resources, the the solutions to the view planning problem are classified as model-based and model-free approaches.

In model-based view planning, a viewing plan is obtained using a previously built or given model of the target [103, 104]. In more general applications and in cases where the target is unknown or introduced to the system in runtime, viewing strategy should be generated without prior information of the target [102]. In such cases, the goal is to manipulate the camera position and orientation in a manner that most

unrevealed information about the target is exposed to the agent in each step. Most of the model-free methods follow the Next-Best-View (NBV) approach. Typically, these systems build an interpretation of the environment using acquired information and base the planning of the next view(s) on it. That interpretation of the scene can be in different forms based on the specific task and application. From this perspective, for the application of 3D reconstruction which is the concern of this work, these methods can be categorized as frontier-based, volumetric-based, or surface-based methods. In frontier-based view planning first introduced in [113], the core idea is to explore unvisited regions of an initial map, i.e. frontiers, to update the map based on the newly collected information in those regions. Methods belonging to this category usually represent the target zone of the environment using a 2D [114, 115] or 3D occupancy grid [105] or directly use a point cloud to map the boundary between explored and unexplored regions [116]. Each pixel or voxel in such grids represents the knowledge of the robot about the presence or absence of an obstacle. As opposed to frontier-based methods that mostly focus on exploring an environment, volumetric-based approaches are usually concerned with modelling a single target and focus more on the completeness of the coverage. In this regard, to guarantee a successful registration, overlaps between the views are also considered while planning the views [117]. In addition to 3D modelling, such algorithms are also used for scene inspection [118, 119]. Some other methods belonging to this category use online partial reconstruction of the map or 3D model without the need for prior knowledge or assumptions about the size or the shape of the target. Accordingly, [120] adopted a probabilistic approach to estimate the information gain from potential viewpoints and plans the views to iteratively reconstruct a 3D model. A subcategory of the model-free view planning algorithms, namely appearance-based planning methods, carries out the decision making process based solely on the visual input such as RGB

or gray-scale images [121, 106, 122]. These methods either rely on an a priori model of the target or a partial reconstruction of the scene, at least during the training stage of their models. For example, a method is proposed in [121] which, based on the information gain from different camera poses, computes a candidate sequence of viewpoints for a micro aerial vehicle to attend. More recently, with the advancements of deep reinforcement learning, appearance-based view planning has been visited more often. Accordingly, [106] utilized only the captured images to plan the views without tracking a partial reconstruction of the true 3D model. They use the surface coverage percentage to guide the agent to propose views that cover the model while minimizing the number of views while doing so. Similarly, [122] used the surface coverage as well as reconstruction error as part of the reward for guiding their reinforcement learning agent towards complete reconstruction. Unlike these methods, the proposed model does not require the true state of the environment or any partial reconstructions for guiding the agent towards capturing high-utility views for the task of reconstruction. In this study, 3D reconstruction is treated as a downstream task rather than a parallel task, and increased attention is given to the conditions that need to be met by the views in order to achieve a satisfactory reconstruction.

It is trivial that as long as coverage is concerned while training the agent, a complete or at least an a priori knowledge of the environment is used for the training of the agent. Although this makes for a guarantee for the maximum coverage of the target, a model of the target is not always accessible for targets newly introduced to the system. On the other hand, compared to the emphasis on the coverage of the target, less attention has been paid to the visual features of the captured views and how they contribute to the reconstruction quality.

4.2.11 Reinforcement Learning Approaches to View Planning

Due to the interactive nature of the view planning problem and its ability to be defined as a sequential decision making problem, reinforcement learning approaches have turned out to be the center of attention to solve this problem in recent years. Meantime, the choice of problem formulation can vary greatly from one work to another based on the specific application and its requirements.

For the task of inspection, [123] has proposed a reinforcement learning-based framework for choosing sub-optimal set of view poses that cover arbitrary 3D models. They use the sensor position and orientation as the agent state which is observed as well as the constructed point cloud based on the previous steps and used the surface area gain of the voxelized point clouds to shape their reward. For satisfying the coverage of target while minimizing the number of required viewpoints, [124] also proposed a reinforcement learning approach to view planning that is needless of tracking a partial online reconstruction of the model to propose the next views and only processes the captured images in runtime. Their state is the preprocessing result of their captured frames through an arbitrary number of frames and they tried both discrete and continuous action space for manipulating the camera poses. Similar to the other work, they used surface coverage to form their reward function. They trained the agent using Deep Q-Network (DQN) and Actor-Critic (AC) algorithms for the discrete and continuous action spaces. Soft Actor-Critic method has also been used in [125] to address the next-best-view planning problem. Their model proposed a sequence of next-best-views to identify the target object.

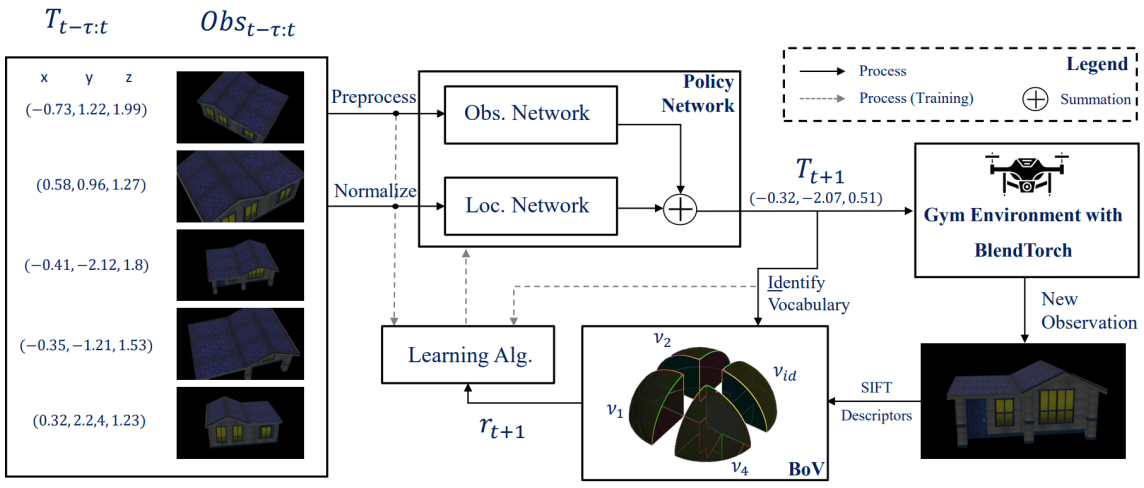

Figure 4.3: Block Diagram of the proposed reinforcement learning-based next-best-view planning .

4.3 Methodology

4.3.1 A Reinforcement Learning Approach to Appearance-based NBV Planning

The problem of NBV planning is a sequential decision making process that can be defined as a Partially Observable Markov Decision Process (POMDP) and be solved through reinforcement learning algorithms. The goal is to achieve this without any need for a priori knowledge of the target and without any full or partial reconstruction of the target during training or inference time.

The goal of the agent in this system is to iteratively propose next views for a limited number of steps to reach regions with a high number of features unfamiliar to the BoV. Seeking such views leads to drastic changes in the vocabularies of the BoV through each relocation of the camera.

The state space should provide the agent with enough information about the environment to enable meaningful actions towards the goal. The input representation utilized consists of the concatenation of down-sampled gray-scale images captured

over the last τ consecutive frames, along with the concatenation of normalized camera locations corresponding to each view. Thus, the state s_t at time t is defined as $\{T_{t-\tau:t}, obs_{t-\tau:t}\}$. The camera location T_t is presented using the spherical coordinate system in the form of $\{R, \phi, \theta\}$ with three values for radial distance from the center, the azimuth angle, and the elevation angle. Also, assuming a deterministic state transition, the action a_t determines the next camera location T_{t+1} after being re-scaled to the specified ranges for its three components. Based on the concept development in Chapter 2, the reward received for this action represents the change in the part of the BoV that has been influenced by the new action, namely $\nu_{T_{t+1}}$ which is the vocabulary associated with the region that the new location belongs to. This change is measured through comparing the same regional vocabulary before and after taking the action; the closest visual words in the two vocabularies are identified and their distance is measured through vector quantization with the cosine distance metric. The sum of these distances is used to represent the change in the vocabulary after taking the action. A negative constant reward is added at each time step, contributing to the overall reward structure of the system:

$$r_{t+1} = dist(\nu_{T_{t+1}}, \nu_{T_t}) - 1 \tag{4.23}$$

where

$$\text{dist}(\nu_{T_{t+1}}, \nu_{T_t}) = \sum_i \left(1 - \cos\left(\nu_{T_{t+1}}(i), \nu_{T_t}(\underset{j}{\text{argmin}} \cos(\nu_{T_{t+1}}(i), \nu_{T_t}(j)))\right)\right) \tag{4.24}$$

and

$$\cos\left(\nu_{T_{t+1}}(i), \nu_{T_t}(j)\right) = \frac{\nu_{T_{t+1}}(i).\nu_{T_t}(j)}{\parallel \nu_{T_{t+1}}(i) \parallel \parallel \nu_{T_t}(j) \parallel} \tag{4.25}$$

Where i iterates over each of visual words in ν_T. The number of vocabularies in

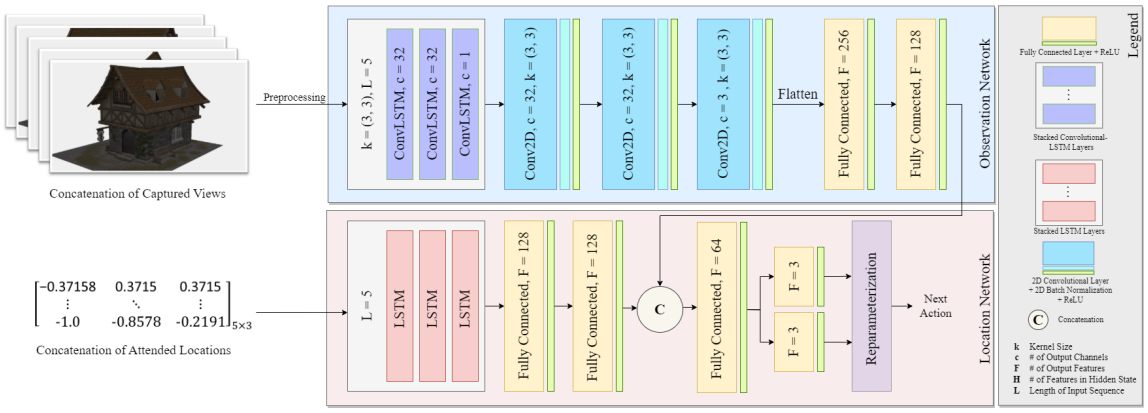

Figure 4.4: Proposed architecture for the policy network.

the BoV and the size of each vocabulary are dependant on the resources available during training and runtime. A demonstration of the workflow for the proposed reinforcement learning-based next-best-view planning is shown in Figure 4.3.

4.3.2 Training Algorithm

To train the agent with the above-mentioned formulation, the Soft Actor-Critic (SAC) algorithm by [108] is used. This is choice is due to several reasons. First, the defined action space is naturally continuous and SAC is of great utility in terms of handling problems with high-dimensional and continuous control tasks. In addition, SAC supports off-policy learning which allows for efficient use of experience samples in the replay buffer. Also, SAC employs maximum entropy reinforcement learning which results in robust and flexible policies enabling high generalization and adaptability.

Beginning the training process, all network weights are initialized with uniformly distributed random values. These weights are the parameters of the actor (policy) network θ_π and the two Q-value (critic) networks ϕ_1 and ϕ_2. Two target critic networks are also initialized with the same parameters as ϕ_1 and ϕ_2. Then, the interactions with the environment begin by taking actions according to the current policy. At each time step t, the new view is captured as a new observation of the state. The state

matrix s_t is formed by concatenating the new view with the last 4 captured views to form a moving window of size 5. The agent then chooses action a_t according to the policy $\pi(s_t)$. This action corresponds to the new location of the camera (R_t, ϕ_t, θ_t), where the camera is oriented to point at the center of the scene. The new state s_{t+1} is then observed which contributes to the calculation of the reward r_t as explained in Section 4.3.1. This transition experience, namely $(s_t, a_t, r_{t+1}, s_{t+1}, d)$ where d is a Boolean indicating whether the termination condition has been reached, is then stored in a replay buffer. This buffer is also called an experience replay [126] which allows for reusing past experiences through a fixed-size buffer of stored samples [127]. In the training phase, a batch of experiences are sampled from this buffer to update the actor and critic networks. This batch contains a diverse range of multiple transitions and prevents bias that could be caused by learning from consecutive experiences [127]. Using the sampled batch, the parameters of the value network, i.e. θ_Q, are learned to minimize the mean squared error between the predicted and target Q-values as explained in Section 4.2.9.

4.3.3 Network Architecture

As shown in Figure 4.3, the policy network is composed of two different sub-networks for processing the two components of the state; the observation network and the location network. The observation network consists of a three-layer convolutional Long Short-Term Memory (LSTM) network followed by three 2D convolutional layers with 32, 32, and 3 channels respectively, all with a 3×3 kernel size. Each convolutional layer is followed by a 2D batch normalization layer and a rectified linear unit (ReLU). The output is then flattened and processed through two fully connected layers of output size 256 and 128. The location network includes the sequence of a three-layer LSTM network with hidden layers of size 128, followed by two fully connected

layers with an output size of 128. The resultant feature vector is then concatenated with the output of the observation network and goes through another fully connected layer with an output size of 128. The final policy results from passing this feature vector through two individual fully connected layers, each with an output size of 3. These layers are responsible for computing the mean and logarithm of the standard deviation of the policy with respect to the input state.

The critic network, which is responsible for mapping state-action pairs to their quality values, encodes the concatenation of the observation component of the state through a sequence of 3D convolutional layers followed by 3D batch normalization layers and ReLU activation functions. Then, the output is flattened and goes through two fully connected layers with an output size of 128. A sub-network also processes the concatenation of the location component of the state and the action through fully connected layers of output size 64 and 128. The resulting feature vector is then concatenated with the encoded observations and goes through two fully connected layers of output size 128 and 3 as the output quality value vector.

4.4 Implementation Details and Experiments

4.4.1 Evaluation of the View Planning Results

To evaluate the utility of the suggested views using the BoV model, the color and depth images were fused into a voxel grid of Truncated Signed Distance Function (TSDF) values following the work in [128]. The mesh and point cloud of this voxel grid were compared with those of the ground truth. In order to have a fair comparison between the point clouds, they should be analyzed through the right similarity metrics. As opposed to the task of 3D reconstruction, where similarity metrics are favored in order to train and evaluate a reconstruction model, the metrics used for

evaluating the upstream view planning task are usually pointed at the completeness of the reconstructed model and viewpoint selection diversity alongside the reconstruction accuracy. The reconstruction results of both the offline dataset refinement and online Next-Best-View (NBV) planning are analyzed using metrics such as Chamfer discrepancy and Hausdorff distance (as explained in Chapter 2). Additionally, a mesh-to-mesh comparison is performed using CloudCompare [23].

Parameter	Value
Batch Size	8
Learning Rate	0.0005
Temperature Parameter (α)	0.2
Target Network Update Interval (steps)	20
Experience Replay Buffer Size	10^6
Soft Update Factor (τ)	0.01
Discount Factor (γ)	0.99

Table 4.1: Hyperparameters used for training the RL agent

4.4.2 Baseline Comparison

Based on the problem formulation in Section 4.3.1 and the training algorithm in Sec 4.3.2, an agent was trained using the hyper-parameters listed on Table 4.1 to scan multiple structures. Figure 4.4.2 shows the primary results of the improvement in the proposed NBVs for two samples from the dataset. Shown in Figure 4.4.2, the range of dissimilarity scores resulting from the BoV method depends on the visual complexity of the target; a higher number of visual features leads to scores with greater magnitudes. In the early stages of training, the agent is prone to following the trend in the change of the viewpoints or suggest views that expose many new features to get a higher reward. However, the reward function takes care of this trade-off and suggests a balanced reward for updating an already existing vocabulary compared to creating a new one. This helps the agent to prioritize taking better looks

at the areas with higher density of unfamiliar features, namely features with higher dissimilarity scores shown with brighter red dots.

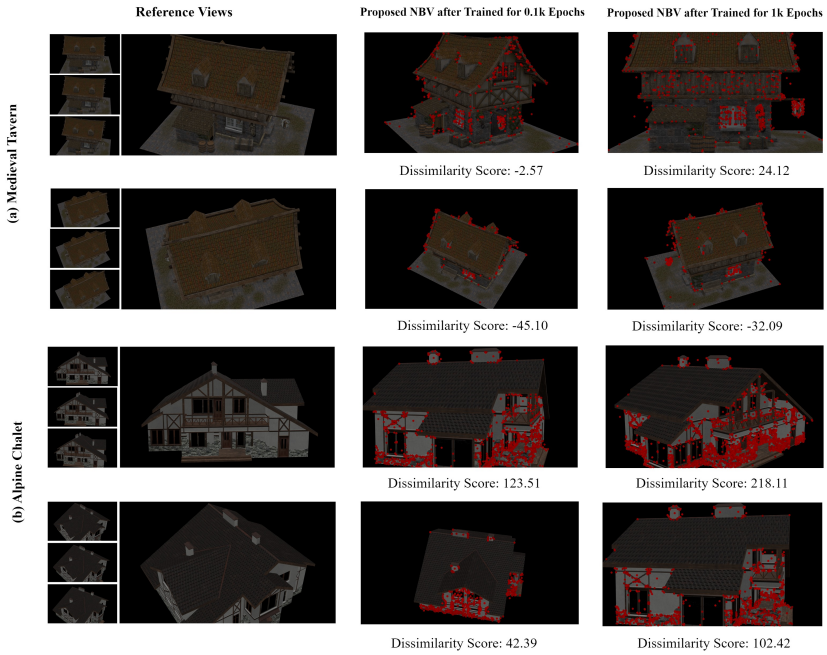

Figure 4.5: Proposed NBV by the trained agent in different training stages for two different environments with the dissimilarity scores used for rewarding the agent. This dissimilarity score is calculated using the BoV method. For initializing the BoV, the last five frames are used which are chosen randomly in the 3D space and are 5 degrees and 2 degrees apart in the azimuth and elevation angles, respectively.

The termination condition was set to be the completion of one pass around the target, highlighting the model's capability to iteratively identify the optimal views. The resultant trajectory for a sample from the dataset and the basline scan are shown in Figure 4.6. Hausdorff distance and Chamfer discrepancy for this reconstruction using Equations 2.8 and 2.9 were calculated to be 2.78 cm and 0.80 cm while the surface coverage was 94.60%.

Furthermore, the policy trained on a single structure (specifically, the *medieval tavern* with medium visual complexity) was employed to assess two unknown buildings (the *imperial temple* with lower complexity and the *alpine chalet* with higher complexity). This comparison aimed to evaluate the results and test the general-

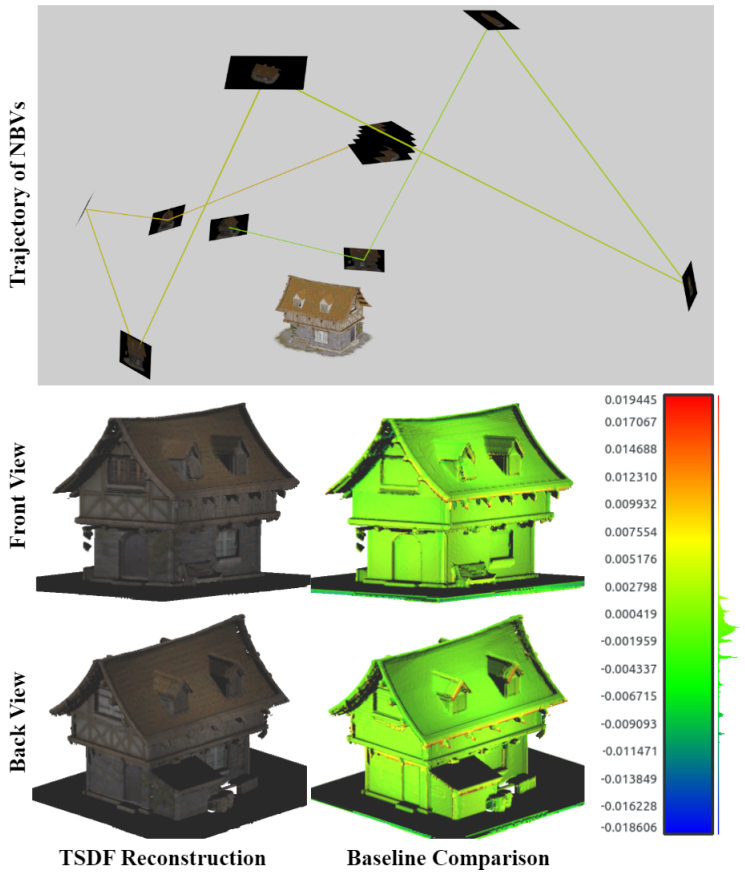

Figure 4.6: Qualitative evaluation of the results of the proposed Bag-of-Views model applied to next-best-view planning. The reconstruction has been compared to the results from a complete coverage baseline scan of the target.

izability of the learned policy. Shown in Figure 4.4.2, the results demonstrate the efficacy of the model in finding the optimal views which result in high-quality reconstructions. Hausdorff distance and Chamfer discrepancy for the *Alpine Chalet* were calculated to be 9.47 cm and 1.01 cm and the same metrics for the *Imperial Temple* were 6.35 cm and 0.82 cm, respectively. The superior performance of the model on lower complexity targets compared to higher complexity targets provides compelling evidence of its proficiency in tracking visual features for guidance. This observation reinforces the model's capability to navigate to the high-utility views for 3D reconstruction.

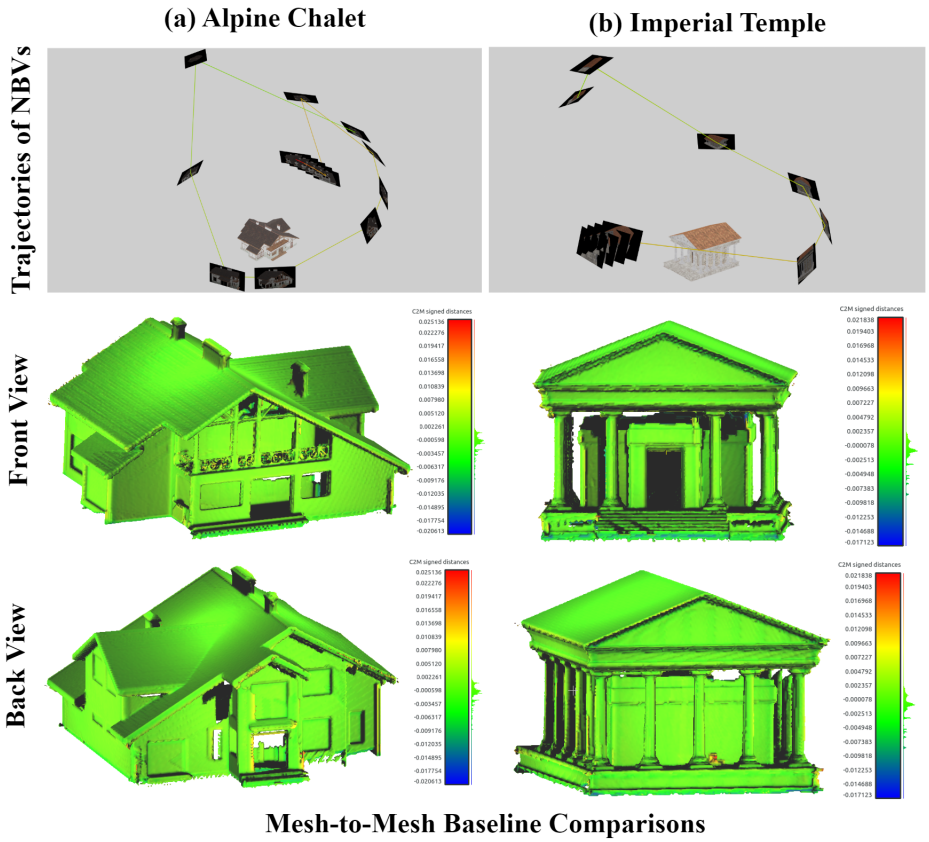

Figure 4.7: Testing the generalizablity of the proposed RL-based NBV planner on two unknown targets.

4.5 Conclusions and Future Work

This study tackled the challenge of model-free view planning by introducing a novel appearance-based computational representation of reconstruction targets. Using the concept developed and tested in Chapter 2, a Bag-of-Views (BoV) model was used to track visual features of the target and manipulate the camera path to capture the most representative views of the target. The application of the Bag-of-Views (BoV) model was extended to modify the reward function of a reinforcement learning (RL) agent trained using the Soft Actor-Critic (SAC) algorithm. This adaptation enabled the RL agent to perform online NBV planning. Once again, the proposed model yielded high-quality reconstructions with a significantly low number of views

(down to 5% of the number of baseline views). Furthermore, the RL model exhibited substantial generalizability to unseen targets. A notable finding of this study was that the level of generalizability depended on the relative visual complexity between the training and testing environments. This observation serves as further validation of the effectiveness of the proposed appearance-based view selection approach. While this work primarily focused on 3D reconstruction, the modular nature of the proposed method lends itself well to customization for various other applications. Promising future research can include pre-training the visual vocabularies of the BoV for tracking certain visual features associated with structural defects in infrastructure.

Chapter 5

Conclusions and Future Work

5.1 Summary

This thesis presented three crucial components of a UAV-based data acquisition system for the task of 3D reconstruction. The main focus of this study was relating the visual features seen in 2D images to the 3D model of the target to avoid the computational cost of handling 3D data. Three main questions rise while developing such a system:

- How can the agent identify which viewpoints are contributing most to the downstream 3D reconstruction task, if not tracking a 3D representation of the agent's knowledge of the scene?
- How can the agent utilize the 2D images to boost the quality of the downstream 3D reconstruction task?
- How can the agent identify the best viewpoints to attend during the data acquisition step?

This study was composed of three main chapters to propose an answer to each of the above-mentioned questions.

Chapter 2 introduced a novel autoencoder-like tool to reconstruct 3D voxel grids and assign utility to each view. By utilizing a multi-view variational autoencoder and a feature-matching method, the best views were selected based on their contribution to the reconstruction. The introduction of a Bag-of-Views (BoV) model reduced the number of views used for reconstruction by 70.6% while decreasing the reconstruction error by 33.5%. The efficacy of the model in identifying optimal views for reconstruction was demonstrated, showcasing its utility in UAV-based aerial photogrammetry.

Chapter 3 focused on harnessing object semantics to enhance the depth completion task, leading to higher density depth images. The improved density of depth maps obtained from this task can subsequently enhance the quality of 3D reconstruction processes. The chapter also addressed the challenges of sensor data defects impacting both 3D modeling and safety insurance in autonomous aerial vehicles. To overcome these challenges, a multi-task network was proposed and implemented to perform depth completion and object detection tasks. Semantic feature maps were introduced as scene representations to improve depth completion results. The network architecture utilized a shared backbone with two task-specific heads. One head focused on generating a dense depth map, while the other localized infrastructure objects by producing bounding boxes in the scene. A comparative analysis was conducted, comparing the results of the single-task and multi-task networks. The findings demonstrated the improved performance of the multi-task network in producing depth maps for pixels belonging to detected objects in the object detection stream.

Chapter 4 tackled the challenge of model-free view planning through the introduction of a novel appearance-based computational representation of reconstruction targets. The innovative Bag-of-Views (BoV) model developed in Chapter 2 was employed to track visual features of the target and manipulate the camera path for capturing the most representative views. Remarkable results were achieved with high-

quality reconstructions using significantly fewer views, often as low as 5% compared to baselines. The RL model demonstrated substantial generalizability to previously unseen targets, dependent on the relative visual complexity of the training and testing environments, validating the effectiveness of the appearance-based view selection approach. The proposed method's modular nature enables customization for various other applications.

5.2 Limitations

One of the objectives of the proposed methods was comprehensibility and modularity while effectiveness in delivering their purposes. The success of each method in simulation environment was confirmed, however, modifications and extra pre-processing and post-processing methods must be added while tackling the real-world environment where lighting and weather conditions pose challenges to feature extraction from acquired 2D images.

Another assumption made in Chapters 2 and 4 was the known position of the agent carrying the camera which was given as a part of the simulation environment. In real-world scenarios, the UAV's pose can be inferred through various technologies. One method is Global Positioning System (GPS) and Inertial Measurement Units (IMUs) that most of the UAVs are equipped with. Integrating these two technologies, the camera pose and orientation can be estimated. Using visual odometry, Ground Control Points (GCPs), and external tracking systems are other methods that can serve this purpose.

5.3 Future Work

The presented research introduced three novel solutions to different components of a UAV-based data acquisition system. Modularity of each of the proposed solutions paves the way for promising improvements and extensions in various ways and fields. Some of the suggested research directions are listed below.

- The mentioned methods in Section 5.2 for localizing the UAV throughout the data acquisition process can introduce uncertainty to the system and thus affecting the view planning results. Handling uncertainty of this matter and extending the BoV model in this regard can ensure more robust planning strategies and avoid confusions while retrieving visual features from spatially defined vocabularies.

- Although the BoV model was based on SIFT features to avoid any dependency on the environment model or context, it can be easily integrated with deep feature extractors that specialize on the class of the specific targets. This can increase the efficiency of the knowledge representation algorithm and the viewing strategy will be more robust to redundant or non-contributing features to the 3D reconstruction task.

- This work can be potentially extended to multi-agent systems for collaborative data acquisition, where multiple UAVs work together to efficiently cover larger areas, gather diverse perspectives, and improve the overall quality of data acquisition.

- Feature explanation and extended multi-level feature fusion for the multi-task network proposed in Chapter 3 can enhance the efficiency and the depth completion accuracy. In addition, experimenting different pre-training modes can potentially help boost the network performance and training time.

Bibliography

[1] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

[2] R. Sutton and A. Barto, "Reinforcement learning: An introduction," *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, vol. 9, p. 1054, 02 1998.

[3] R. W. Beard and T. W. McLain, *Small unmanned aircraft: Theory and practice*. Princeton university press, 2012.

[4] K. Nonami, F. Kendoul, S. Suzuki, W. Wang, and D. Nakazawa, *Autonomous flying robots: unmanned aerial vehicles and micro aerial vehicles*. Springer Science & Business Media, 2010.

[5] R. K. Barnhart, D. M. Marshall, and E. Shappee, *Introduction to unmanned aircraft systems*. Crc Press, 2021.

[6] R. Szeliski, *Computer vision: algorithms and applications*. Springer Nature, 2022.

[7] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[8] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4104–4113, 2016.

[9] G. Hu, Z. Zhou, J. Cao, and H. Huang, "Highly accurate 3d reconstruction based on a precise and robust binocular camera calibration method," *IET Image Processing*, vol. 14, no. 14, pp. 3588–3595, 2020.

[10] X. Wang, X. Luo, M. Yu, C. Shi, Y. Zhu, and C. Gong, "A framework for 3d reconstruction dataset preprocessing," in *2019 International Conference on Virtual Reality and Visualization (ICVRV)*, pp. 251–252, 2019.

[11] X. X. Zhu, D. Tuia, L. Mou, G.-S. Xia, L. Zhang, F. Xu, and F. Fraundorfer, "Deep learning in remote sensing: A comprehensive review and list of resources," *IEEE Geoscience and Remote Sensing Magazine*, vol. 5, no. 4, pp. 8–36, 2017.

[12] X.-F. Han, H. Laga, and M. Bennamoun, "Image-based 3d object reconstruction: State-of-the-art and trends in the deep learning era," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 5, pp. 1578–1604, 2021.

[13] J. Shang, T. Shen, S. Li, L. Zhou, M. Zhen, T. Fang, and L. Quan, "Self-supervised monocular 3d face reconstruction by occlusion-aware multi-view geometry consistency," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV*, pp. 53–70, Springer, 2020.

[14] B. O. Community, *Blender 3.0 - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2022.

[15] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," vol. 2, pp. 2169 – 2178, 02 2006.

[16] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on statistical learning in computer vision, ECCV*, vol. 1, pp. 1–2, Prague, 2004.

[17] M. Aharchi and M. Ait Kbir, "A review on 3d reconstruction techniques from 2d images," in *Innovations in Smart Cities Applications Edition 3: The Proceedings of the 4th International Conference on Smart City Applications 4*, pp. 510–522, Springer, 2020.

[18] O. Özyeşil, V. Voroninski, R. Basri, and A. Singer, "A survey of structure from motion.," *Acta Numerica*, vol. 26, p. 305–364, 2017.

[19] H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, no. 5828, pp. 133–135, 1981.

[20] K. Shervais, "Structure from motion introductory guide," *Version Oct*, vol. 22, p. 2015, 2015.

[21] AgiSoft, "AgiSoft PhotoScan Professional (Version 1.2.6)." Software, 2016*. Retrieved from http://www.agisoft.com/downloads/installer/.

[22] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "MeshLab: an Open-Source Mesh Processing Tool," in *Eurographics Italian Chapter Conference* (V. Scarano, R. D. Chiara, and U. Erra, eds.), The Eurographics Association, 2008.

[23] C. O. Community, "Cloudcompare - 3d point cloud and mesh processing software open source project," 2022.

[24] Z. Gao, E. Li, Z. Wang, G. Yang, J. Lu, B. Ouyang, D. Xu, and Z. Liang, "Object reconstruction based on attentive recurrent network from single and multiple images," *Neural Processing Letters*, vol. 53, no. 1, pp. 653–670, 2021.

[25] Z. Liu, W. Qv, H. Cai, H. Guan, and S. Zhang, "An efficient and robust hybrid sfm method for large-scale scenes," *Remote Sensing*, vol. 15, no. 3, 2023.

[26] Q. Mi and T. Gao, "3d reconstruction based on the depth image: A review," in *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pp. 172–183, Springer, 2022.

[27] K.-H. Chang, "Chapter 2 - geometric modeling," in *e-Design* (K.-H. Chang, ed.), pp. 41–124, Boston: Academic Press, 2015.

[28] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611, pp. 586–606, Spie, 1992.

[29] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr, "Implicit fairing of irregular meshes using diffusion and curvature flow," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 317–324, 1999.

[30] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE international symposium on mixed and augmented reality*, pp. 127–136, Ieee, 2011.

[31] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *ACM siggraph computer graphics*, vol. 21, no. 4, pp. 163–169, 1987.

[32] Z. Zhu, X. Wang, S. Bai, C. Yao, and X. Bai, "Deep learning representation using autoencoder for 3d shape retrieval," vol. 204, pp. 41–50.

[33] H. Xie, H. Yao, X. Sun, S. Zhou, and S. Zhang, "Pix2vox: Context-aware 3d reconstruction from single and multi-view images," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2690–2698, 2019.

[34] H. Xie, H. Yao, S. Zhang, S. Zhou, and W. Sun, "Pix2vox++: multi-scale context-aware 3d object reconstruction from single and multiple images," *International Journal of Computer Vision*, vol. 128, no. 12, pp. 2919–2935, 2020.

[35] G. Fahim, K. Amin, and S. Zarif, "Single-view 3d reconstruction: A survey of deep learning methods," *Computers & Graphics*, vol. 94, pp. 164–190, 2021.

[36] G. Yang, Y. Cui, S. Belongie, and B. Hariharan, "Learning single-view 3d reconstruction with limited pose supervision," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 86–101, 2018.

[37] X. Li, S. Liu, K. Kim, S. De Mello, V. Jampani, M.-H. Yang, and J. Kautz, "Self-supervised single-view 3d reconstruction via semantic consistency," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*, pp. 677–693, Springer, 2020.

[38] J. Hartung, P. M. Dold, A. Jahn, and M. Heizmann, "Analysis of ai-based single-view 3d reconstruction methods for an industrial application," *Sensors*, vol. 22, no. 17, 2022.

[39] M. Gadelha, S. Maji, and R. Wang, "3d shape induction from 2d views of multiple objects," in *2017 International Conference on 3D Vision (3DV)*, pp. 402–411, IEEE, 2017.

[40] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th annual international conference on machine learning*, pp. 609–616, 2009.

[41] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Multi-view 3d models from single images with a convolutional network," in *European Conference on Computer Vision*, pp. 322–337, Springer, 2016.

[42] J. Wu, T. Xue, J. J. Lim, Y. Tian, J. B. Tenenbaum, A. Torralba, and W. T. Freeman, "Single image 3d interpreter network," in *European Conference on Computer Vision*, pp. 365–382, Springer, 2016.

[43] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," *arXiv preprint cs/0205070*, 2002.

[44] A. Bosch, X. Munoz, and R. Marti, "Which is the best way to organize/classify images by content?," *Image and vision computing*, vol. 25, no. 6, pp. 778–791, 2007.

[45] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[46] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

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

[48] L. Fei-Fei and P. Perona, "A bayesian hierarchical model for learning natural scene categories," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, pp. 524–531, IEEE, 2005.

[49] K. Arai and A. Barakbah, "Hierarchical k-means: An algorithm for centroids initialization for k-means," *Reports of the Faculty of Science and Engineering*, vol. 36, pp. 25–31, 01 2007.

[50] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 2169–2178, 2006.

[51] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, pp. 2564–2571, 2011.

[52] M. Tucsok, S. H. Gazani, K. Gupta, and H. Najjaran, "3d reconstruction from 2d images: A two-part autoencoder-like tool," in *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 538–543, 2022.

[53] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 2161–2168, 2006.

[54] A. Angeli, D. Filliat, S. Doncieux, and J.-A. Meyer, "Fast and incremental method for loop-closure detection using bags of visual words," *IEEE transactions on robotics*, vol. 24, no. 5, pp. 1027–1037, 2008.

[55] R. Yu, C. Long, G. Ma, J. Guo, L. Xu, and Z. Guo, "An improved deep-learning monocular visual slam method based on local features," in *Fifth International*

Conference on Computer Information Science and Artificial Intelligence (CISAI 2022), vol. 12566, pp. 1086–1100, SPIE, 2023.

[56] J. Krantz, T. Gervet, K. Yadav, A. Wang, C. Paxton, R. Mottaghi, D. Batra, J. Malik, S. Lee, and D. Chaplot, "Navigating to objects specified by images," 04 2023.

[57] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

[58] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction," in *European conference on computer vision*, pp. 628–644, Springer, 2016.

[59] A. Lucieer, S. De Jong, and D. Turner, "Mapping landslide displacements using structure from motion (sfm) and image correlation of multi-temporal uav photography," *Progress in Physical Geography*, vol. 38, pp. 97–116, 02 2013.

[60] H.-T. Zhang, J. Yu, and Z. Wang, "Probability contour guided depth map inpainting and superresolution using non-local total generalized variation," *Multimedia Tools and Applications*, vol. 77, 04 2018.

[61] A. Eldesokey, M. Felsberg, K. Holmquist, and M. Persson, "Uncertainty-aware cnns for depth completion: Uncertainty from beginning to end," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12014–12023, 2020.

[62] L. Yan, K. Liu, and E. Belyaev, "Revisiting sparsity invariant convolution: A network for image guided depth completion," *IEEE Access*, vol. 8, pp. 126323–126332, 2020.

[63] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," *Proceedings of The 33rd International Conference on Machine Learning*, 06 2015.

[64] J. Hu, C. Bao, M. Ozay, C. Fan, Q. Gao, H. Liu, and T. L. Lam, "Deep depth completion from extremely sparse data: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[65] Y. Zuo, Y. Fang, Y. Yang, X. Shang, and Q. Wu, "Depth map enhancement by revisiting multi-scale intensity guidance within coarse-to-fine stages," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 12, pp. 4676–4687, 2020.

[66] S. Yan, C. Wu, L. Wang, F. Xu, L. An, K. Guo, and Y. Liu, "Ddrnet: Depth map denoising and refinement for consumer depth cameras using cascaded cnns," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 151–167, 2018.

[67] Y.-J. Chang, S. Kim, and Y.-S. Ho, "Depth upsampling methods for high resolution depth map," in *2018 International Conference on Electronics, Information, and Communication (ICEIC)*, pp. 1–4, 2018.

[68] M. A. U. Khan, D. Nazir, A. Pagani, H. Mokayed, M. Liwicki, D. Stricker, and M. Z. Afzal, "A comprehensive survey of depth completion approaches," *Sensors*, vol. 22, no. 18, p. 6969, 2022.

[69] L. Bai, Y. Zhao, M. Elhousni, and X. Huang, "Depthnet: Real-time lidar point cloud depth completion for autonomous vehicles," *IEEE Access*, vol. 8, pp. 227825–227833, 2020.

[70] A. Eldesokey, M. Felsberg, K. Holmquist, and M. Persson, "Uncertainty-aware cnns for depth completion: Uncertainty from beginning to end," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12014–12023, 2020.

[71] A. Eldesokey, M. Felsberg, and F. S. Khan, "Propagating confidences through cnns for sparse data regression," *arXiv preprint arXiv:1805.11913*, 2018.

[72] Y. Xu, X. Zhu, J. Shi, G. Zhang, H. Bao, and H. Li, "Depth completion from sparse lidar data with depth-normal constraints," pp. 2811–2820, 10 2019.

[73] J. Qiu, Z. Cui, Y. Zhang, X. Zhang, S. Liu, B. Zeng, and M. Pollefeys, "Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image," pp. 3308–3317, 06 2019.

[74] Y.-K. Huang, T.-H. Wu, Y.-C. Liu, and W. Hsu, "Indoor depth completion with boundary consistency and self-attention," pp. 1070–1078, 10 2019.

[75] Y. Zhang and T. Funkhouser, "Deep depth completion of a single rgb-d image," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 175–185, 2018.

[76] F. Ma and S. Karaman, "Sparse-to-dense: Depth prediction from sparse depth samples and a single image," 2018.

[77] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[78] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee, "A survey of modern deep learning based object detection models," *Digital Signal Processing*, p. 103514, 2022.

[79] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1, pp. 886–893, Ieee, 2005.

[80] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, pp. 303–338, 2010.

[81] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740–755, Springer, 2014.

[82] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, *et al.*, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *International Journal of Computer Vision*, vol. 128, no. 7, pp. 1956–1981, 2020.

[83] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proceedings of the IEEE*, vol. 111, no. 3, pp. 257–276, 2023.

[84] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[85] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[86] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[87] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[88] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, pp. 41–75, 1997.

[89] S. Ruder, "An overview of multi-task learning in deep neural networks," *arXiv preprint arXiv:1706.05098*, 2017.

[90] L. Duong, T. Cohn, S. Bird, and P. Cook, "Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser," in *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (volume 2: short papers)*, pp. 845–850, 2015.

[91] T. Standley, A. Zamir, D. Chen, L. Guibas, J. Malik, and S. Savarese, "Which tasks should be learned together in multi-task learning?," in *International Conference on Machine Learning*, pp. 9120–9132, PMLR, 2020.

[92] W. Bailer and H. Fassold, "Resource-efficient object detection by sharing backbone cnns," in *2019 IEEE International Symposium on Multimedia (ISM)*, pp. 196–1963, IEEE, 2019.

[93] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun, "Multi-task multi-sensor fusion for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7345–7353, 2019.

[94] N. Zou, Z. Xiang, Y. Chen, S. Chen, and C. Qiao, "Simultaneous semantic segmentation and depth completion with constraint of boundary," *Sensors*, vol. 20, no. 3, p. 635, 2020.

[95] Y.-K. Huang, T.-H. Wu, Y.-C. Liu, and W. Hsu, "Indoor depth completion with boundary consistency and self-attention," pp. 1070–1078, 10 2019.

[96] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2117–2125, 2017.

[97] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," *Proceedings of The 33rd International Conference on Machine Learning*, 06 2015.

[98] T. maintainers and contributors, "TorchVision: PyTorch's Computer Vision library," Nov. 2016.

[99] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.

[100] W. Falcon and The PyTorch Lightning team, "PyTorch Lightning," Mar. 2019.

[101] L. Teixeira, M. R. Oswald, M. Pollefeys, and M. Chli, "Aerial single-view depth completion with image-guided uncertainty estimation," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1055–1062, 2020.

[102] S. Chen, Y. Li, and N. M. Kwok, "Active vision in robotic systems: A survey of recent developments," *The International Journal of Robotics Research*, vol. 30, no. 11, pp. 1343–1377, 2011.

[103] W. R. Scott, "Model-based view planning," *Machine Vision and Applications*, vol. 20, no. 1, pp. 47–69, 2009.

[104] A. Aryan, F. Bosché, and P. Tang, "Planning for terrestrial laser scanning in construction: A review," *Automation in Construction*, vol. 125, p. 103551, 2021.

[105] A. Kleiner and C. Dornhege, "A frontier-void-based approach for autonomous exploration in 3d," *Advanced Robotics*, vol. 27, 11 2011.

[106] D. Peralta, J. Casimiro, A. Nilles, J. Aguilar, R. Atienza, and R. Cajote, *Next-Best View Policy for 3D Reconstruction*, pp. 558–573. 01 2020.

[107] A. Torralba, A. Oliva, M. S. Castelhano, and J. M. Henderson, "Contextual guidance of eye movements and attention in real-world scenes: the role of global features in object search.," *Psychological review*, vol. 113, no. 4, p. 766, 2006.

[108] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, pp. 1861–1870, PMLR, 2018.

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

[110] M. Morales, "Grokking deep reinforcement learning," 2020.

[111] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[112] "Soft actor-critic." https://spinningup.openai.com/en/latest/algorithms/sac.html. Accessed: 2022-12-18.

[113] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, pp. 146–151, 1997.

[114] E. Vidal, N. Palomeras, K. Istenič, J. D. Hernández, and M. Carreras, "Two-dimensional frontier-based viewpoint generation for exploring and mapping underwater environments," *Sensors*, vol. 19, no. 6, 2019.

[115] S. Ahmad, A. Mills, E. Rush, E. Frew, and J. Humbert, "3d reactive control and frontier-based exploration for unstructured environments," pp. 2289–2296, 09 2021.

[116] S. Manandhar and R. Sadananda, "Effect of hamming distance of patterns on storage capacity of hopfield network," pp. 253– 256 vol.1, 12 2002.

[117] J. Vasquez-Gomez, L. Sucar, and R. Murrieta-Cid, "View/state planning for three-dimensional object reconstruction under uncertainty," *Autonomous Robots*, vol. 41, 01 2017.

[118] S. Song and S. Jo, "Online inspection path planning for autonomous 3d modeling using a micro-aerial vehicle," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6217–6224, 2017.

[119] L. Heng, A. Gotovos, A. Krause, and M. Pollefeys, "Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1071–1078, 2015.

[120] J. Daudelin and M. Campbell, "An adaptable, probabilistic, next-best view algorithm for reconstruction of unknown 3-d objects," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1540–1547, 2017.

[121] C. Forster, M. Pizzoli, and D. Scaramuzza, "Appearance-based active, monocular, dense reconstruction for micro aerial vehicles," 07 2014.

[122] S. Potapova, A. Artemov, S. Sviridov, D. Musatkina, D. Zorin, and E. Burnaev, "Next best view planning via reinforcement learning for scanning of arbitrary 3d shapes," *Journal of Communications Technology and Electronics*, vol. 65, pp. 1484–1490, 2020.

[123] C. Landgraf, B. Meese, M. Pabst, G. Martius, and M. F. Huber, "A reinforcement learning approach to view planning for automated inspection tasks," *Sensors*, vol. 21, no. 6, 2021.

[124] D. Peralta, J. Casimiro, A. Nilles, J. Aguilar, R. Atienza, and R. Cajote, *Next-Best View Policy for 3D Reconstruction*, pp. 558–573. 01 2020.

[125] C. Korbach, M. Solbach, R. Memmesheimer, D. Paulus, and J. Tsotsos, "Next-best-view estimation based on deep reinforcement learning for active object classification," 10 2021.

[126] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine learning*, vol. 8, pp. 293–321, 1992.

[127] W. Fedus, P. Ramachandran, R. Agarwal, Y. Bengio, H. Larochelle, M. Rowland, and W. Dabney, "Revisiting fundamentals of experience replay," in *International Conference on Machine Learning*, pp. 3061–3071, PMLR, 2020.

[128] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3dmatch: Learning local geometric descriptors from rgb-d reconstructions," in *CVPR*, 2017.

Appendix A

Sample Dataset Refinement Results

In the following, some of the most significant samples from Figure 2.9 are included for improved readability. These figures demonstrate the reconstruction outcomes achieved using various sizes of Bag-of-Visual-Words (BoV) alongside different vocabulary dimensions for two samples from the dataset. It is important to note that while error values are presented, they do not holistically capture the actual quality of the reconstructed model.

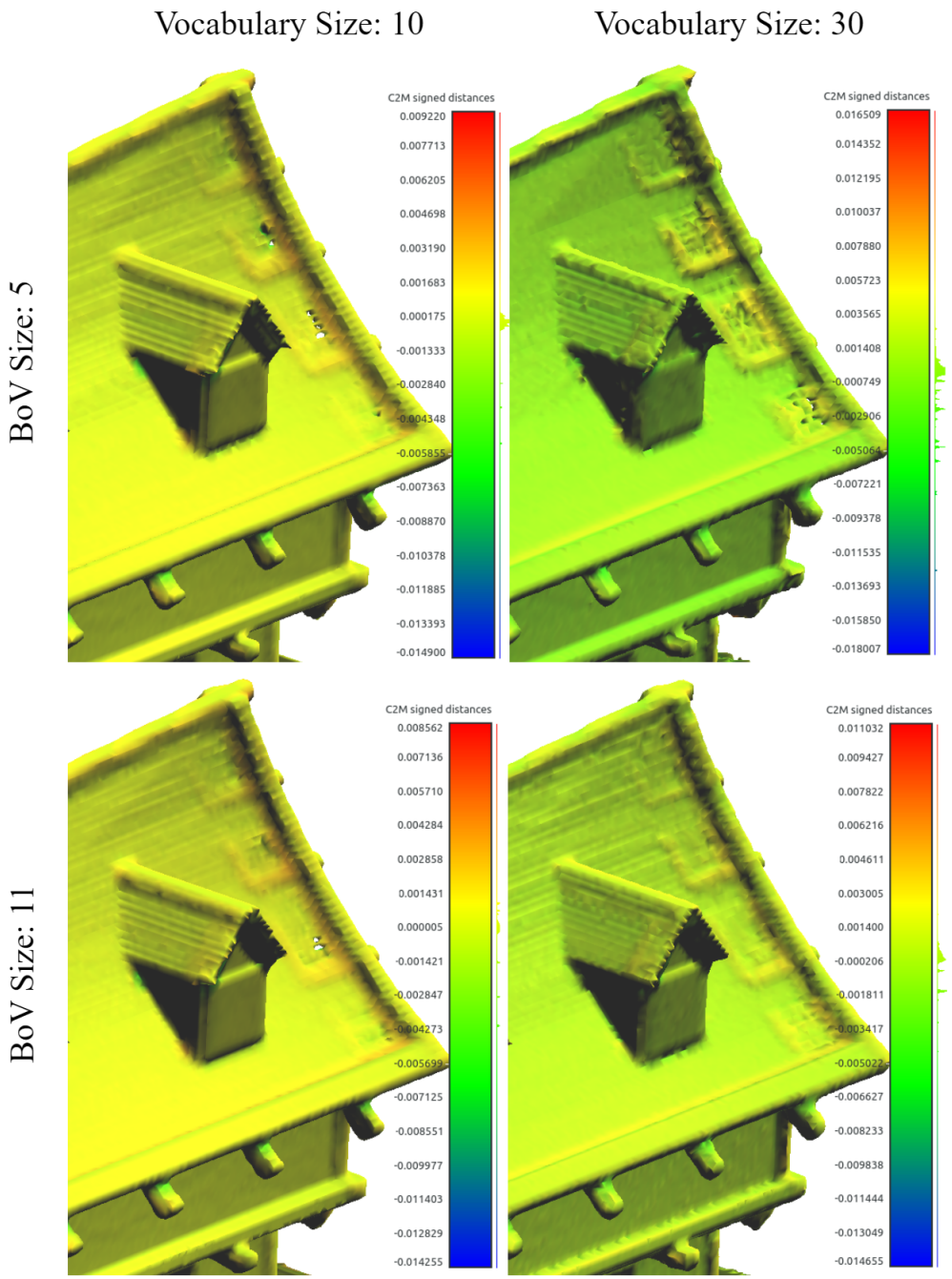

Figure A.1: Sample results for *Medieval Tavern*.

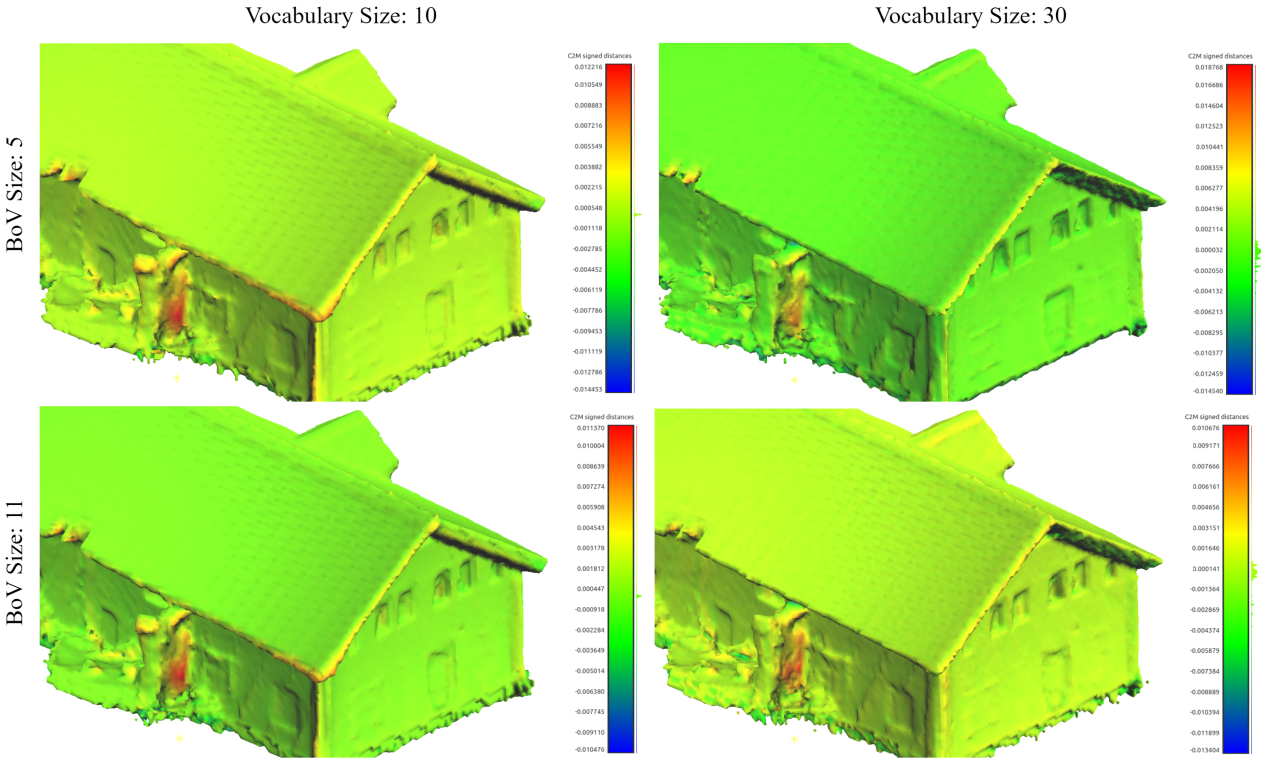

Figure A.2: Sample results for *House Ruin*.