Ecole Polytechnique
Federale de Lausanne

Eidgenössische Technische
Hochschule Zürich

Master Thesis

VIRUS-NeRF

Vision, InfraRed, and UltraSonic based
Neural Radiance Fields for local mapping

Author:
Nicolaj Schmid

Supervisors:
Cornelius von Einem
Florian Tschopp
Lorenz Hruby

LABs:
Autonomous Systems Lab
Predictive Control Lab

Professors:
Roland Siegwart
Colin Jones

Start:
12.08.2023

Finish:
09.02.2024

Contact:
nicolaj.schmid@epfl.ch

Version:
V1 (09.02.2024)

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

VIRUS-NeRF - Vision, InfraRed, and UltraSonic based Neural Radiance Fields for local mapping

Authored by (in block letters):

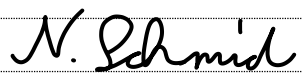
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Schmid	Nicolaj

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Olten, 09.02.2024	

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Contents

Acknowledgments

I had a lot of insightful revelations during my master's thesis: For example, sensor calibration does take more than one day. Normally, GPU crashes are not someone else's fault but my own. And most importantly, there is still a lot more to discover in the field of robotics and I am looking forward to it.

I am deeply grateful to you, Cornelius von Einem, Florian Tschopp and Lorenz Hruby, for your warm support. You not only introduced me to the exciting world of science but also showed me lots of practical skills. Even after the closure of your team, Florian and Lorenz, you continued to give me precious advice and share your extensive knowledge. Cornelius, your consistent assistance and words of encouragement guided me through many challenges. Additionally, I am thankful to Roland Siegwart for making the project possible at ASL ETHZ and to Colin Jones for supervising it and facilitating connections with EPFL. Finally, I thank my family, my friends and my girlfriend for being involved in many deep talks about robots and for distracting me when the NeRFs got on my nerves. It has been an absolute pleasure collaborating with each of you. Thank you!

Abstract

This research strives to leverage cost-effective sensors for local mapping applications in mobile robotics. The study introduces *VIRUS-NeRF - Vision, InfraRed, and Ultra-Sonic based Neural Radiance Fields*. While traditional mapping techniques often rely on expensive sensors like LiDARs or RGB-D cameras, *VIRUS-NeRF* integrates low-cost sensors while using *Neural Radiance Fields* (NeRFs) for scene representation. By exhaustive sensor analysis and testing, the URM37 could be identified as the optimal *UltraSonic Sensor* (USS). Building upon *Instant Neural Graphics Primitives* (*Instant-NGP*), *VIRUS-NeRF* incorporates depth measurements from USSs and *InfraRed Sensors* (IRSs) and advances the occupancy grid utilized for ray marching. Experimental evaluation conducted at ETHZ demonstrates that *VIRUS-NeRF* achieves comparable mapping performance to LiDAR point clouds in terms of coverage, and surpasses USS and IRS scans. Notably, in environments with optimized parameters, its accuracy aligns with that of LiDAR measurements, while in less optimized settings, it exhibits performance akin to USSs. Through an in-depth ablation study, three key factors of NeRF-based mapping are identified:

1. Utilizing *Instant-NGP* for mapping yields poor results. The assistance of the camera with depth sensors is imperative given the sparse measurements typical in mobile robotics.

2. The proposed occupancy grid in *VIRUS-NeRF* augments mapping performance and training efficiency.

3. *Particle Swarm Optimization* (PSO) of the large hyper-parameter space and refinement of poses through bundle adjustment enhances accuracy.

Limitations such as accuracy constraints, hyper-parameter generalization issues, and convergence speed are discussed and accompanied by possible solutions. Overall, *VIRUS-NeRF* presents a promising approach for cost-effective local mapping in mobile robotics, with potential applications in safety and navigation tasks. The code is found on *Github* under https://github.com/ethz-asl/virus_nerf.

Keywords: local mapping, NeRF, implicit neural representation, Instant-NGP, occupancy grid, low-cost sensors, infrared sensor, ultrasonic sensor, camera

Symbols

Symbols & Indices

NeRF

i	enumerate pixels/rays, $i = 1, ..., N$
j	enumerate samples, $j = 1, ..., M$
$\mathbf{C}_i$	RGB ground truth colour of ray/pixel
$\hat{\mathbf{C}}_i$	RGB colour prediction of ray/pixel
$\hat{\mathbf{c}}_j$	RGB colour prediction of sample
D_i	ground truth depth of ray/pixel
$\hat{D}_i$	depth prediction of ray/pixel
d_j	depth of sample
ω_j	weight of sample
σ_j	density prediction of sample
δ_j	distance between adjacent samples
T_j	light transmittance at sample
$\mathcal{L}_{tot}$	total loss
$\mathcal{L}_c$	colour loss
$\mathcal{L}_{USS}$	ultrasonic sensor loss
$\mathcal{L}_{IRS}$	infrared sensor loss

Occupancy Grid

i	enumerate cells
P	occupancy grid probability
c_i	cell i, $c_i = occ/emp$ cell is occupied/empty
M_n	measurement n
P_F	false detection probability
σ_i	density prediction of cell i
σ_{Tmax}	fixed density threshold

Acronyms and Abbreviations

ASL	Autonomous Systems Lab
EM	Expectation Maximization
FoV	Field of Vies
GPU	Graphics Processing Unit
GT	Ground Truth
I2C	Inter-Integrated Circuit
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
IRS	InfraRed Sensor
MAE	Mean Absolute Error
MDE	Monocular Depth Estimation
MLP	Multi-Layer Perceptron
NGP	Neural Graphics Primitives
NND	Nearest Neighbour Distance
PCB	Printed Circuit Board
PSNR	Peak Signal to Noise Ratio
PSO	Particle Swarm Optimization
PWM	Pulse Width Modulation
RGB-D	Red Green Blue Depth
RMSE	Root Mean Squared Error
ROS	Robotic Operating System
SFM	Structure From Motion
SLAM	Simultaneous Localization and Mapping
SMB	Super Mega Bot
ToF	Time of Flight
USS	UltraSonic Sensor

Chapter 1

Introduction

This research aims to utilize cost-effective sensors for local mapping applications in mobile robotics. The study evolved from a collaboration between the *Autonomous Systems Lab* (ASL) at ETHZ and *Arrival* having a particular interest in indoor environments and safety-related applications. Unlike global mapping, which captures entire scenes, local mapping focuses on the immediate vicinity of the robot. Local representations prove to be effective for tasks such as obstacle circumnavigation and collision avoidance due to their fast updating speeds and lightweight memory structures. In this work, local maps are studied in the simplified case of static environments.

Early mapping algorithms often utilized inexpensive sensors such as *UltraSonic Sensors* (USS) [1, 2, 3], and *InfraRed Sensors* (IRS). Their performance was limited by low resolution, sparse sampling or short range. Contemporary approaches frequently rely on more advanced sensors like *Light Detecting And Ranging* (LiDAR) sensors or depth cameras. Despite their impressive performance, these setups incur substantial costs, making them less accessible for widespread adoption. Recent advancements in stereo vision [4, 5], visual *Simultaneous Localization And Mapping* (SLAM) [6] and *Monocular Depth Estimation* (MDE) [7, 8, 9] show significant progress in camera-based mapping solutions. However, the intrinsic scale ambiguity of images, the lack of robustness and high computational requirements remain open research challenges. Sensor fusion may address some of these drawbacks, e.g. depth completion [10, 11, 12], but most times, these techniques use again costly LiDAR or RGB-D sensors.

This work presents *Vision, InfraRed and UltraSonic based Neural Radiance Fields* (*VIRUS-NeRF*) for local mapping. *VIRUS-NeRF* pursues the core idea of fusing cameras and low-cost depth sensors investigated in two prior semester projects [13, 14]. In addition to USSs and cameras, IRSs are incorporated into the sensor setup. Instead of employing depth completion based on deep learning, the sensor measurements are fused inside the framework of *Neural Radiance Fields* (NeRFs) [15] learning an implicit neural representation of the environment. Similar to other works using LiDARs [16, 17] or depth cameras [18, 19], the image-based training is complemented by USS and IRS depth measurements. Notably, *VIRUS-NeRF* is the first NeRF algorithm utilizing low-cost depth sensors.

Chapter 2 introduces the current state of the art of mapping techniques. Given the inherent noise in cheap sensors, careful device selection is crucial. In chapter 3, the sensor arrangement is discussed and the optimal USS is selected based on extensive market research followed by in-depth testing of three sensors. Chapter 4 presents *VIRUS-NeRF*, subsequently assessed in real-world experiments detailed in chapter 5. The experimental process includes dataset collection and optimization of poses and hyper-parameters. The local mapping performance is evaluated against instantaneous scans of USSs, IRSs and LiDARs. The study concludes in chapter 6, where potential future enhancements and extensions are deliberated.

Chapter 2

Literature Review

2.1 Overview

In this chapter, the relevant literature on sensor fusion and mapping is presented. Each mapping algorithm is largely dependent on the sensors of the robot. Most importantly, this project aims to use cost-efficient sensors which is discussed in chapter 3. This requirement, along with an emphasis on local mapping and a strong interest in learning-based methods, confines the scope of related work taken into consideration.

In the context of ranging sensors, occupancy grids are one of the most established mapping techniques. They are a probabilistic framework presented in chapter 2.2. Then, camera-based depth estimation namely *Monocular Depth Estimation* (MDE) and depth completion are reviewed in chapter 2.3. In chapter 2.4, *Neural Radiance Fields* (NeRFs) are introduced: a relatively new approach to learning scene representations. In chapter 2.5, the various mapping and fusion methods are discussed in the context of this project and the *VIRUS-NeRF* approach is justified. For a general introduction to map representations including OctoMaps, Signed-Distance-Fields and Surfel-Maps, the reader is referred to annexe A.1.

2.2 Occupancy Grids

Moravec and Elfes introduce the occupancy grid: a probabilistic map describing the occupancy state of a discretised environment [1]. Many of the early publications explicitly use USSs to create occupancy grids [1, 20, 21, 22]. In the original implementation, the environment is represented by two sub-maps containing the probability of each cell being empty or occupied respectively. By each new measurement M_n, the sub-maps are updated using the probabilistic addition rule: $P(c_i|M_1, ..., M_n) = P(c_i|M_1, ..., M_{n-1}) + P(c_i|M_n) - P(c_i|M_1, ..., M_{n-1})P(c_i|M_n)$ where $c_i = emp$ or $c_i = occ$. This updating step is used during the prior semester project about sensor fusion of USS [14]. Keeping track of two sub-maps, as in the primary occupancy grid, is cumbersome. Therefore, the probability addition rule is replaced by the more elegant Bayesian updating rule [2] (see chapter 4.3.2 for the mathematical details).

To reduce the updating complexity, the Bayesian occupancy grid considers consecutive measurements and neighbouring cells to be independent. The *MURIEL* method addresses the correlation between successive samples by joining measurements into pose bins according to their position and viewing direction [21]. Moreover, *MURIEL* uses a USS model considering multiple reflections and multiple targets. The assumption of spatial independence can be dropped by solving the mapping problem in high-dimensional space using expectation maximization (EM) [22]. The previously presented methods are compared in an empirical study using USSs [23]: The EM technique produces the

most accurate maps at a high computational cost. The best real-time algorithm is the *MURIEL* method.

Multiple works extend occupancy grids to handle dynamical objects: The *Bayesian Occupancy Filter* uses a 4D occupancy grid containing information about the speed of the objects [24]. Other approaches use *Particle Filters* [25] or *Markov Chains* [26, 27] to work in dynamical environments.

2.3 Camera Depth Estimation

2.3.1 Monocular Depth Estimation

Occupancy grids and similar scene representations rely on depth measurements, yet cameras capture only RGB images, posing an ill-posed problem due to the lack of spatial context. Nevertheless, recent advances in deep learning have led to surprisingly good results in *Monocular Depth Estimation* (MDE).

Deep learning-based MDE can be divided into three categories: supervised, unsupervised and semi-supervised learning [8]. In supervised learning, the loss between the output of the model and the ground truth is used for training. Most times, the ground truth is obtained by LiDAR sensors. Getting a good ground truth is complicated because the point cloud of a LiDAR sensor is much more sparse than camera images. Therefore, unsupervised learning is introduced relying only on a video stream of unlabelled images [28]. The method utilises consecutive images of the same scene (I_1, I_2) that have slightly different perspectives. A subsequent image (I_2) can be warped to an approximation of a previous one ($\tilde{I}_1$) by using a depth estimation D_1: $I_2(D_1) \rightarrow \tilde{I}_1$. Then, the loss between I_1 and $\tilde{I}_1$ is used for training. Semi-supervised learning uses labelled and unlabelled data.

The KITTI dataset [29] is the most popular benchmark for depth estimation. It gathers outdoor data from driving scenes and contains over 44k images. The best results are achieved by supervised learning and have an RMSE of about 2.3m [30] (see [7, 8] for comparison, status as of mid-2022). However, most of these networks are relatively large and therefore not suited for real-time applications. Some methods achieve better interference time but make trade-offs in accuracy [31]. Another limitation is that MDE does not model uncertainty which is required for many mapping techniques, e.g. occupancy grids. Multiple approaches to estimating the uncertainty are presented in annexe A.2.

2.3.2 Depth Completion

While cameras provide dense information, they suffer from scale ambiguity. Other sensors provide highly accurate but sparse depth measurements. Depth completion tries to leverage the fusion of RGB images with depth measurements. Almost all depth completion applications are based on deep learning and complement the camera with a LiDAR sensor. The measurements can be fused at different levels of the model [11]: In early fusion, the measurements are converted to a unified representation and fused already after a few layers. In late fusion, the measurements are processed separately and fused at the end. Furthermore, there are intermediate solutions and sequential models. The latter concatenates MDE and LiDAR fusion models. The best depth completion model [32] achieves an RMSE of approximately 0.7m on the KITTI benchmark (see [10, 11, 12] for comparisons, status as of mid-2022). Its architecture is a spatial propagation network which is based on affinity matrices forcing spatial correlations between neighbouring pixels [10].

As an alternative to LiDAR point clouds, radar scans can be used for depth completion [33, 34, 35]. These models are relatively new because the resolution of radar sensors

improved in the past few years [34] and *nuScenes*, an automotive dataset containing radar data, was published in 2020 [36]. Existing LiDAR methods do not apply well to radars because the data is much more sparse, it is more noisy and it has a smaller vertical *Field of View* (FoV) [33, 35]. Depth completion based on USSs would be a low-cost alternative to LiDARs and radars. To the best of our knowledge, there are no published papers on sensor fusion between a camera and USS except for the prior semester project [13].

2.4 Neural Radiance Fields

2.4.1 General Concept

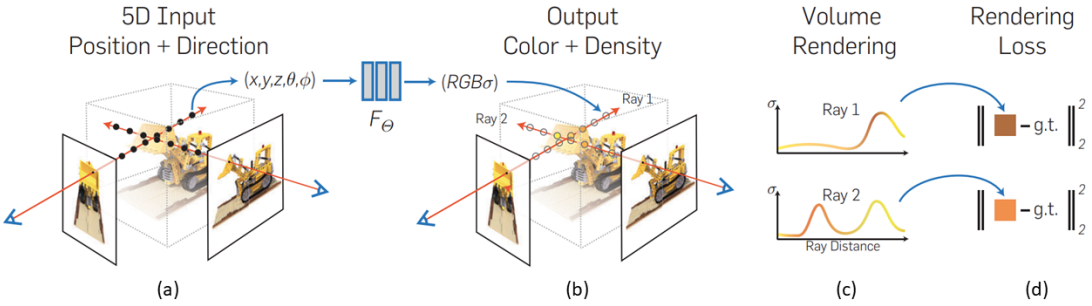


Figure 2.1: NeRF training process (Mildenhall et al. [15])

In 2020, Mildenhall et al. introduce *Neural Radiance Fields* (NeRF) [15]. NeRF use *Multi-Layer Perceptrons* (MLPs) to learn the geometry and the lighting of three-dimensional static scenes. The training data is composed of images and their camera poses. In contrast to most deep learning applications, the model overfits the data to represent as closely as possible one particular environment. During interference, the model can render a new view from any position and viewing direction. The training process is shown in figure 2.1: During ray marching, M pairs of positions and viewing directions are sampled along a ray of a particular pixel i (step (a) in figure 2.1). These samples are input to an MLP that is only a few layers deep. The network outputs the estimated colour ($\hat{\mathbf{c}}_j$) and density (σ_j) of each sample on the ray (step (b) in figure 2.1). Then, through volume rendering, the actual colour of the ray ($\hat{\mathbf{C}}_i$) is estimated (step (c) in figure 2.1):

$$\hat{\mathbf{C}}_i = \sum_{j=1}^{M} T_j (1 - e^{-\sigma_j \delta_j}) \hat{\mathbf{c}}_j \tag{2.1}$$

where δ_j is the distance between adjacent samples and T_j is the light transmittance:

$$T_j = exp(-\sum_{l=1}^{j-1} \sigma_l \delta_l) \tag{2.2}$$

Finally, the squared error between all estimated ray colours ($\hat{\mathbf{C}}_i$) and the corresponding pixels ($\mathbf{C}_i$) is calculated (step (d) in figure 2.1). This loss ($\mathcal{L}_c$) is used for backpropagation:

$$\mathcal{L}_c = \sum_{i=1}^{N} ||\hat{\mathbf{C}}_i - \mathbf{C}_i||_2^2 \tag{2.3}$$

The original NeRF implementation [15] has some important limitations: The training of one scene takes a long time, often over 12h on a GPU (see table 1 in [37]). In addition, NeRFs are designed for small scenes, i.e. single objects or small rooms, and they struggle with unbounded environments [17]. Dense and structured images are required for training [38] and surfaces can be rugged due to a lack of geometrical constraints [39]. Many of these drawbacks are addressed in subsequent publications as presented in the next section.

2.4.2 Improving Works

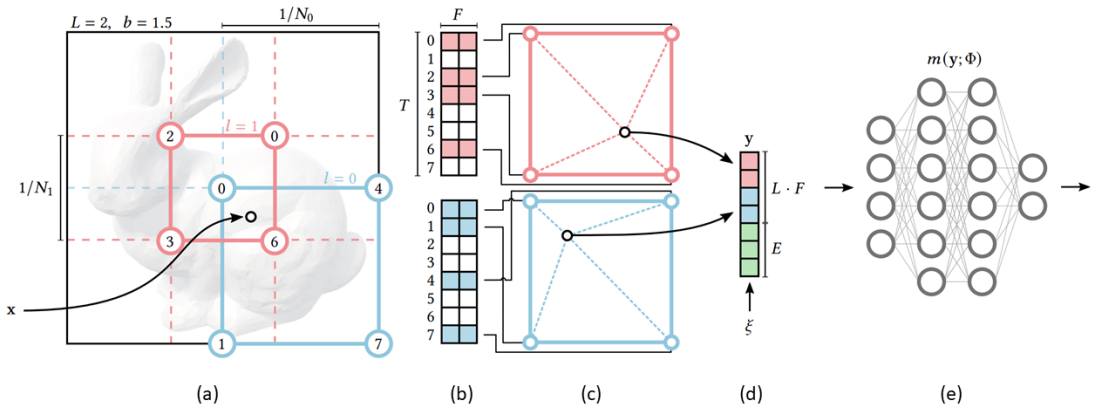


Figure 2.2: Hash encoding of *Instant-NGP* (Müller et al. [40])

Instant Neural Graphics Primitives with a Multiresolution Hash Encoding (*Instant-NGP*) reduces the training time from several hours to a few minutes while achieving a similar accuracy [40]. Instead of a sinusoidal encoding like in the original implementation, *Instant-NGP* uses a multi-resolution hash encoding: First, the input is discretized by hashing the voxel vertices for L resolution levels, i.e. sampling the corner indices for L grids with varying resolutions (step a in figure 2.2). With the use of a spatial hash function [41] the high-dimensional input is mapped to a fixed-sized vector (step b in figure 2.2). Then, the resulting corner indices are linearly interpolated (step c in figure 2.2), concatenated (step d in figure 2.2) and passed through the neural network (step e in figure 2.2). During training, the loss is back-propagated through the neural network and the linear interpolation, all the way to the hash table. In addition, *Instant-NGP* proposes to use a 128^3 occupancy grid for ray-marching. The grid is updated every 16 training iterations by the following heuristic rule:

1. Decay the occupancy grid by a factor of 0.95.
2. Randomly sample some voxels and set their occupancy to the maximum of the density prediction and the previous occupancy.

The resulting occupancy grid is thresholded by a fixed value. During ray-marching all cells which are indicated to be empty by the thresholded occupancy grid are skipped, therefore improving the sampling efficiency.

MegaNeRF [42] and *BlockNeRF* [43] explore the scalability of NeRFs. Using drones, *MegaNeRF* performs large-scale reconstruction of urban environments. It splits the environment into multiple sub-parts and trains corresponding sub-modules in parallel. *BlockNeRF* pushes the scalability even further by performing city-scale reconstructions. It is trained on 2.8 million street-view images.

NeRF in the Wild [38], *Urban Radiance Fields* [16] and *CLONeR* [17] address challenging unbounded and dynamic outdoor scenes. *Urban Radiance Fields* and *CLONeR* use LiDAR point clouds in addition to camera images. *NeRF in the Wild* considers image variations e.g. exposure, lighting, weather and post-processing, and it decomposes the scene into static and transient components. *CLONeR* separates the occupancy and the colour into two MLPs where the occupancy MLP is trained with LiDAR point clouds and the colour MLP with images.

UNISURF [44], *NeuS* (Neural Surface) [39] and *NeuRIS* [45] optimize the surface reconstruction of NeRFs. *NeuS* utilizes signed distance functions (see chapter A.1) achieving better results than *UNISURF* while *NeuRIS* [45] makes use of the surface normal and works for large textureless surfaces. The current cutting edge in terms of surface reconstruction (including large outdoor scenes) is *Neuralangelo* [46]. *Neuralangelo* is based on *Instant-NGP* optimizing the hash grid with a coarse-to-fine approach. Similar to *NeuS*, *Neuralangelo* is based on signed distance functions adding some smoothing constraints.

Additional algorithms investigate NeRFs within the framework of SLAM. *iMAP* [18] and *NICE-SLAM* [19] employ depth supervision from an RGB-D camera. Utilizing GPU acceleration, *iMAP* achieves real-time performance at 10Hz for tracking and updates the map every 2Hz. Unlike conventional NeRF methods, *iMAP* omits the incorporation of viewing direction since specularities are less relevant in SLAM applications. *NICE-SLAM* enhances *iMAP* by introducing a hierarchical scene representation akin to the multi-level encoding of *Instant-NGP*. On the other hand, *NeRF-SLAM* [47] and *Orbeez-SLAM* [48] utilize monocular cameras by separating pose estimation from neural scene representation and leveraging visual odometry. Notably, *NeRF-SLAM* outperforms *iMAP* in terms of depth and colour predictions while exhibiting comparable depth predictions and superior colour predictions compared to *NICE-SLAM*. *NeRF-SLAM* uses *Droid-SLAM* [49] as a tracking module and *Instant-NGP* for scene representation.

2.5 Algorithm Selection

How can the above-presented mapping and fusion algorithms be useful in the context of this project? One simple approach is to concatenate existing methods that are suited for low-cost sensors. For example, depth images could be estimated by MDE and then fused in an occupancy grid with other depth sensors. The simplicity of such an approach is appealing. However, late fusion inside an occupancy grid makes it likely that any sensor weakness is directly propagated into the map instead of being compensated by the strengths of other sensors. Ineffective sensor fusion of low-cost devices is expected to result in inaccurate mapping and to be not robust.

Alternatively, the depth completion from the prior semester project using cameras and USSs could be pursued and extended to IRSs. This approach is abandoned due to two reasons: First, The previous semester's project shows that artificial datasets can be used for training, but the resulting network is struggling to generalize in new environments [13]. An even larger error has to be expected from the simulation-to-reality gap concluding that a real-world dataset is required. However, no dataset exists containing camera, USS and IRS measurements and creating one, which is large enough for deep learning, is a difficult and time-consuming task. Second, the best MDE has an RMSE of 2.3m on the *KITTI* benchmark [30]. Depth completion with a LiDAR improves the accuracy to 0.7m being only about three times better than MDE [32]. USSs and IRSs have more sparse and more noisy data compared to LiDARs and it is not obvious why depth completion using these sensors should bring any improvement over MDE.

NeRFs present a novel framework for implicit scene representation being the preferred choice for this project due to the following reasons:

- By adding multiple losses and back-propagating them into the neural map, NeRFs have an interface to fuse colour images with depth measurements. Hence, NeRFs address the sensor fusion and the map representation in one algorithm.
- In contrast to many mapping techniques, NeRFs are continuous and not discrete which allows a higher resolution.
- Saving a map inside an MLP is memory efficient. In our experiments, the smallest room with a volume of about $130\,\mathrm{m}^3$ is represented by less than 32 MB (including MLP, hash tables and occupancy grid). Comparably, a 3D occupancy grid with $1\,\mathrm{cm}^3$ resolution is more than 16 times larger.
- Besides occupancy, NeRFs learn colour and lighting properties which could be used for other tasks, e.g. object classification or segmentation.

VIRUS-NeRF is based on *Instant-NGP* [40] because this is a state-of-the-art model considering its fast convergence speed and its wide adaption [46, 47, 50, 51]. Notably, *NeRF-SLAM* [47] uses the algorithm in the context of mapping. Moreover, *Instant-NGP* utilizes an occupancy grid for efficient ray-marching. This is a well-established concept when using USSs which can be capitalized as described in chapter 4.3.

Chapter 3

Sensors

3.1 Overview

This chapter is dedicated to the development of the sensor arrangement. The sensor setup is a fundamental layer of any mapping algorithm. In chapter 3.2.1, sensor requirements are defined based on the goal of the project followed by extensive USS market research in chapter 3.2.2. Then, the sensor setup of *VIRUS-NeRF* is established in chapter 3.2.3. Consider reading chapter B.1 for an in-depth analysis of USS discussing its physical properties, advantages and limitations. Further, some alternative depth sensors, namely LiDARs and stereo vision, are presented in chapters B.2 and B.3. In any multi-sensor setup, cross-talking may be problematic. The interference between two pairs of USSs and IRSs is assessed by two qualitative experiments, in chapter B.4. In addition, some strategies for avoiding cross-talk between USSs are examined.

Low-cost sensors may have large differences in performance and particularly USS are susceptible to faults (see chapter B.1). This makes it all the more important to test multiple sensors and to understand the characteristics of the sensors including the opening angle, the accuracy and the measurement noise level. Based on the USS market research three sensors are selected and these sensors are evaluated by two experiments in chapter 3.3. The IRSs are added at a later stage of the project and they are tested only qualitatively.

3.2 Sensor Arrangement

3.2.1 Requirements

The sensor requirements are defined according to the project aiming for local mapping (1) of indoor environments (2-3) using low-cost sensors (4-5).

1. The sensors should have a range of at least 3 meters.
2. The coverage of the sensors should be as broad as possible for safe operations, i.e. collision avoidance.
3. The robot is operating at slow speeds (maximum $1\,\mathrm{m\,s^{-1}}$) and a minimal sampling frequency of 5Hz is sufficient.
4. Low-cost sensors are defined as being below 50 CHF such that the entire sensor setup does not exceed a few hundred Swiss francs.
5. The sensors must be available in the lab or at a retailer operating in Switzerland and the shipping time should not exceed 4 weeks.

3.2.2 USS Market Search

Table 3.1 shows the result of the market search and contains 14 USS sensors from 7 companies. Most companies operate mainly in the robotic industry. USSs used in other industries (e.g. car industry) seem to be difficult to integrate.

3.2.3 Sensor Selection

Table B.1 shows a qualitative comparison of different ranging sensors and guides the sensor selection. Limiting the setup to cheap sensors eliminates the use of LiDARs or assisted stereo vision, e.g. *RealSense D455* (see chapter B.2 and B.3). All cheap sensors are narrowly limited in performance. For example, USSs have a poor angular resolution and cameras have scale ambiguity. *VIRUS-NeRF* suggests fusing USSs and cameras combining good coverage with high resolution. Inspired by *NanoSLAM* [52], *VL53L5CX* IRSs are added to the arrangement. As shown in chapter 5.5.1, IRS have very good accuracy and they transfer this feature to *VIRUS-NeRF*.

The *VL53L5CX* from *STMicroelectronics* [53] is an 8x8 multi-zone *Time of Flight* (ToF) IRS sensor with a FoV of 45°. The sensor has a range of 4 m and costs only around 7.5 CHF. The *RealSense D455* [54] is taken as a camera. Nevertheless, the algorithm relies only on colour images and any cheaper camera could be used instead. The depth images of the *D455* are uniquely used for debugging and comparison. Based on the USS market search, the following three sensors are ordered and tested:

- HC-SR04: It is one of the cheapest USSs (4 CHF). The sensor is often used in low-budget robotic applications, for example, the prior semester project [14].
- URM37: This USS has a wide range (2-800 cm) and incorporates temperature compensation. In terms of price, it is in the middle of the analysed sensors (13 CHF).
- MB1603: This is one of the more expensive sensors (38 CHF). However, it has two modes that are made for a multi-sensor system: First, in *sensor chaining* one sensor triggers directly the next one once the measurement has finished. Secondly, the sensors have a cross-talk rejection which should allow multi-sensor setups without interference.

The sensor CH201 is ordered as well because contrary to the other ones it is based on MEMS technology. However, the piece gets lost during the shipping process and is not tested.

Sensor	Company	Price [CHF]	Availability	Vcc [V]	Current [mA]	Temperature [°C]	Communication	link
HC-SR04	SparkFun	4	in stock	5	15	-	echo	SparkFun
Grove	Seed Studio	4	in stock	3.2-5.2	8	-10 to 60	echo	Seed Studio
US-100	Adafruit	7	in stock	2.4-5.5	2	-20 to 70	echo or serial	Adafruit
URM37	DFRobot	13	in stock	3.3-5.5	20	-10 to 70	serial, pulse width or analogue	DFRobot
JSN-SR04T	DFRobot	15	in stock	3-5.5	8	-20 to 70	serial or pulse width	DFRobot
EV_MOD CH101-01-02	TDK InvenSense	15	in stock	1.8	0.05	-40 to 85	serial	TDK
EV_MOD CH201-00-01	TDK InvenSense	23	in stock	1.8	0.25	-40 to 85	serial	TDK
URM13	DFRobot	23	4 weeks	3.3-5.5	-	-10 to 70	serial	DFRobot
SRF05	DFRobot	28	4 weeks	5	30	-	pulse width	DFRobot
PING	Parallax	32	in stock	5	30	0 to 70	pulse width	Parallax
MB1000	MaxBotix	32	in stock	2.5-5.5	2	-40 to 65	serial, pulse width or analogue	MaxBotix
MB1033	MaxBotix	38	2 weeks	2.5-5.5	3.1	0 to 65	serial, pulse width or analogue	MaxBotix
MB1603	MaxBotix	38	2 weeks	2.5-5.5	3.5	-40 to 65	serial, pulse width, analogue	MaxBotix
SRF08	DFRobot	46	4 weeks	5	15	-	serial	DFRobot

Sensor	Range [cm]	Resolution [cm]	Error [%]	Angle [°] (-6dB)	Sound freq. [kHz]	Sample freq. [Hz]	Features
HC-SR04	2-400	approx. 0.3	-	30	40	15	-
Grove	2-350	1	-	15	40	-	-
US-100	2-450	0.3	1	15	-	-	temperature measurement in UART mode
URM37	2-800	1	1	-	-	10	temperature measurement or compensation, servo-motor controlling
JSN-SR04T	21-600	0.3	-	75	40	-	waterproof sensor (separate PCB + 2m cable)
EV_MOD CH101-01-02	4-120	0.1	1mm RMS (at 30cm)	45	175	100	very small (8x5x5mm)
EV_MOD CH201-00-01	20-500	0.1	-	45	85	25 (5m), 100 (1m)	very small (8x8x6mm)
URM 13	15-900	1	1	60	40	10	2 modes: 15-150cm (50Hz) and 40-900cm (10Hz)
SRF05	1-400	-	-	60	40	20	-
PING	3-300	-	-	approx. 40	40	-	-
MB1000	15-645	2.54	-	different models	42	20	sensor chaining
MB1033	30-500	0.1	1	different models	42	10	cross-talk rejection, temperature compensation, sensor chaining
MB1603	2-500	0.1	1	different models	42	10	cross-talk rejection, temperature compensation, sensor chaining
SRF08	3-600	-	-	60	40	15	change signal gain to lower measurement period

Table 3.1: USS market research: three price categories: 0-10 CHF in dark orange, 10-30 CHF in light orange and 30-50 CHF in yellow (according to DigiKey Switzerland, 09.08.2023). The communication type can be either *echo* (trigger response as soon a sound wave is measured), *serial* (serial communication like UART or I2C), *pulse width* (the length of the pulse indicates the distance) or *analogue* (the voltage level indicates the distance). The feature *sensor chaining* means that one USS can trigger the next one via a direct electrical connection.

3.3 Ultrasonic Sensor Experiments

3.3.1 Set-up

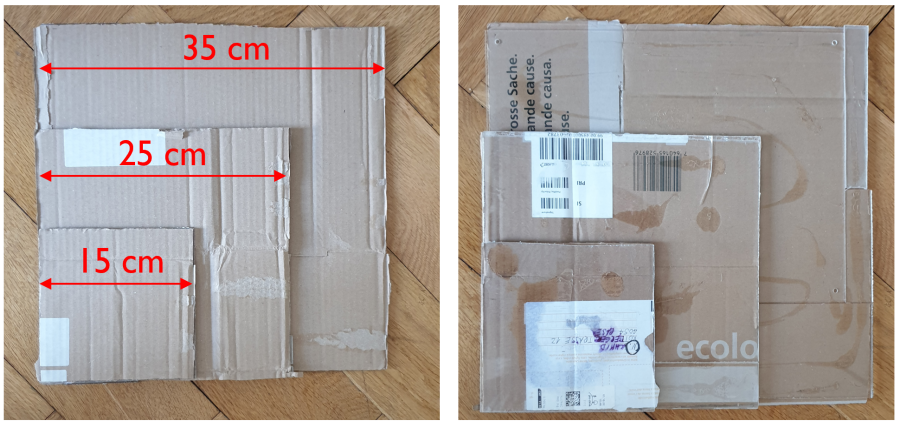
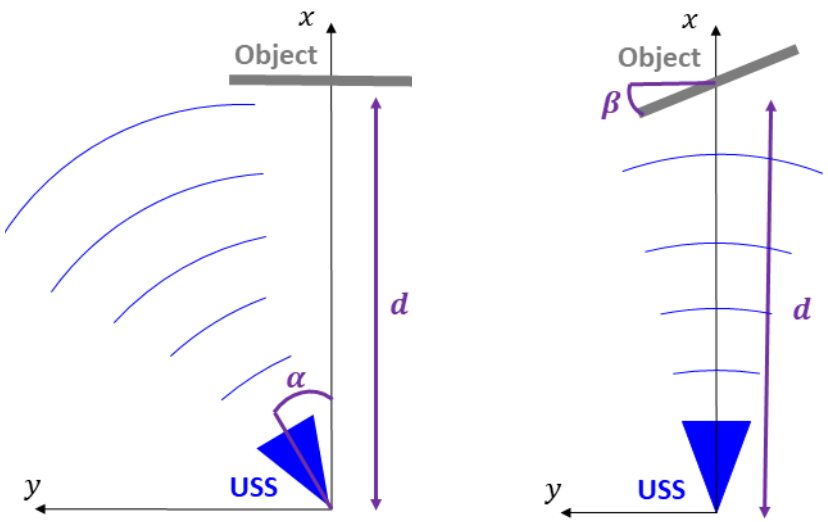


Figure 3.1: Objects used for the USS experiments: three sizes and double-sided material having cardboard on one side (left) and plexiglass on the other (right)

The ranging capabilities of the USSs are tested by measuring the distance to six objects. The objects are squares in three sizes (small = 15 cm^2, medium = 25 cm^2 and large = 35 cm^2) and they are made out of cardboard or plexiglass (see figure 3.1). The objects are attached to a microphone stand and placed in an empty room to make sure that nothing but the object of interest reflects the sound waves. The microphone stand is round and small (1.5 cm in diameter) such that it reflects only a few sound waves to the sensor. Testing shows that the stand can only trigger USSs at a distance below 50 cm.

Two distinct experiments are conducted: In the first experiment, the USS is inclined to the viewing direction by varying the angle from $\alpha = -40°$ to $\alpha = 40°$ in steps of 10° (see figure 3.2a). The sound waves hit the surface of the object in parallel. In the second experiment, the obstacle is inclined to the viewing direction from $\beta = 0°$ to $\beta = 90°$ in steps of 22.5° and the waves hit the object only in parallel if $\beta = 0°$ (see figure 3.2b). In the first experiment, the sensors are tested at a distance of 0.25 m, 0.5 m, 1 m and 2 m while the second experiment evaluates only distances of 1 m and 2 m for the medium-sized objects (25 cm^2). For every sensor-object configuration, the distance measurement is repeated 200 times. Then, the *Mean Absolute Error* (MAE) and the standard deviation between the measurements and the *Ground Truth* (GT) are calculated.



(a) Experiment 1 (b) Experiment 2

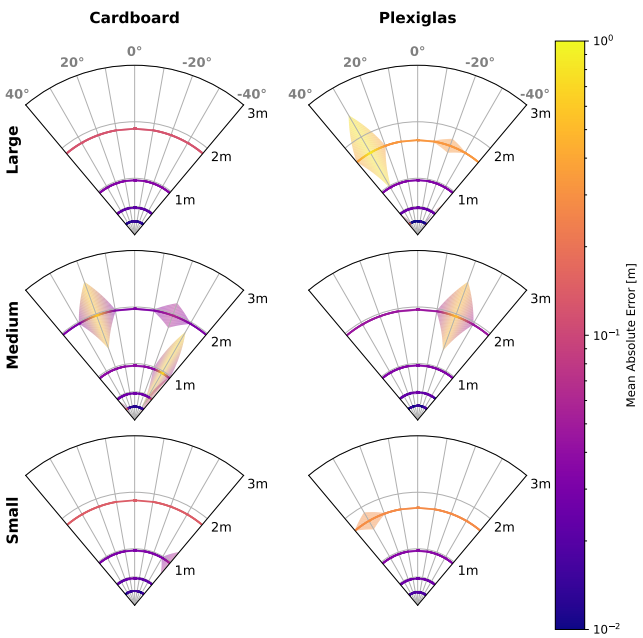
Figure 3.2: USS experiments setup: In the first experiment, the orientation of the USS (α) is varied and in the second one, the normal direction of the flat object (β).

3.3.2 Results

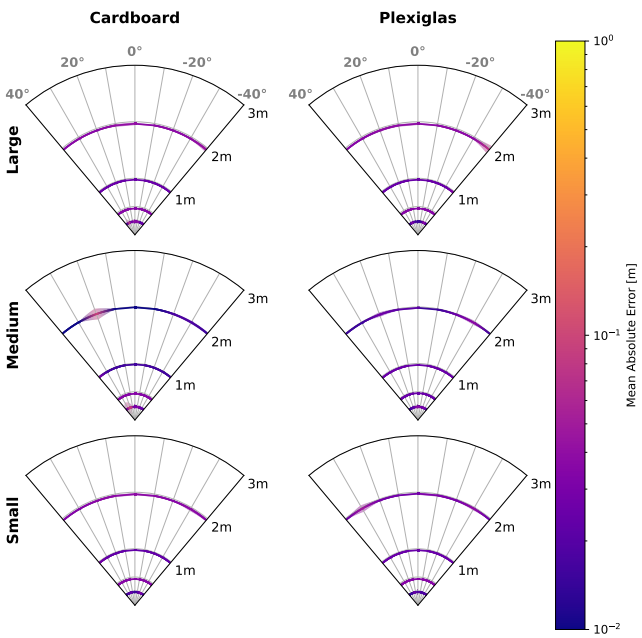
The results of the two experiments are represented in figure 3.3. The first experiment shows that the accuracy and the precision decrease for larger distances for the sensors *HC-SR04* and *MB1603* while the *URM37* seems to be less affected. For 1 m, all USSs detect objects reliably. At 2 m the sensors *HC-SR04* and *MB1603* have a large standard deviation for some inclination angles due to not detecting the object. In general, the FoV can be considered to be at least 80° if the surface of the object is parallel to the sound wave. The second experiment shows that if the object is not in parallel to the wave ($\beta > 0°$ in figure 3.2), then the FoV decreases rapidly. In particular, the sensor *HC-SR04* does not detect any object at a distance of 2 m if $\beta \geq 22.5°$. For the sensors *URM37* and *MB1603*, the objects out of plexiglass are slightly harder to detect than objects out of cardboard, probably because the flatter surface of plexiglass reflects the wave while the rougher texture of cardboard scatters it more in random directions. Particularly for the cardboard object, the *URM37* makes more stable measurements than the *MB1603*.

3.3.3 Discussion

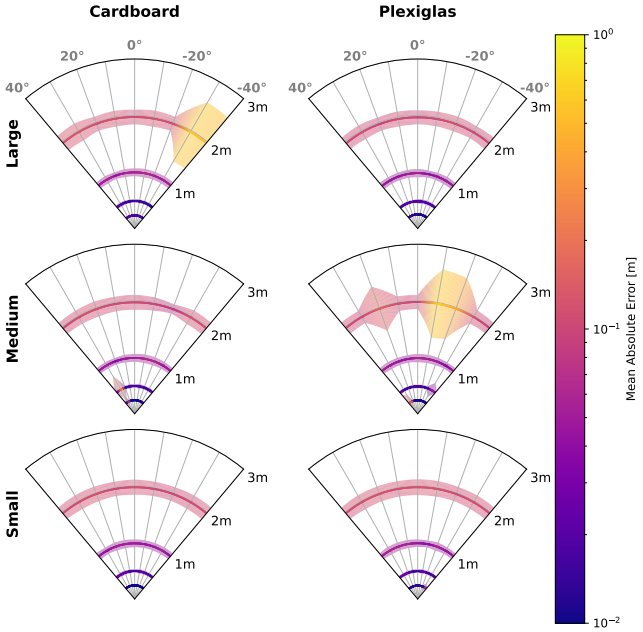
In both experiments, the *URM37* performs the best and therefore, it is used for *VIRUS-NeRF*. The *MB1603* has worse accuracy and detection properties compared to the *URM37* while costing about three times more (38 CHF instead of 13 CHF). The *URM37* still costs about three times more than the *HC-SR04* (4 CHF), but this money is a good investment considering the significantly better results of *URM37*. Qualitative cross-talking experiments are done after ordering multiple *URM37* devices (see chapter B.4). For applications that are sensitive to sensor interference, e.g. multi-robot systems, it could be interesting to test similarly multiple *MB1603* sensors which promise to have a good cross-talk rejection.



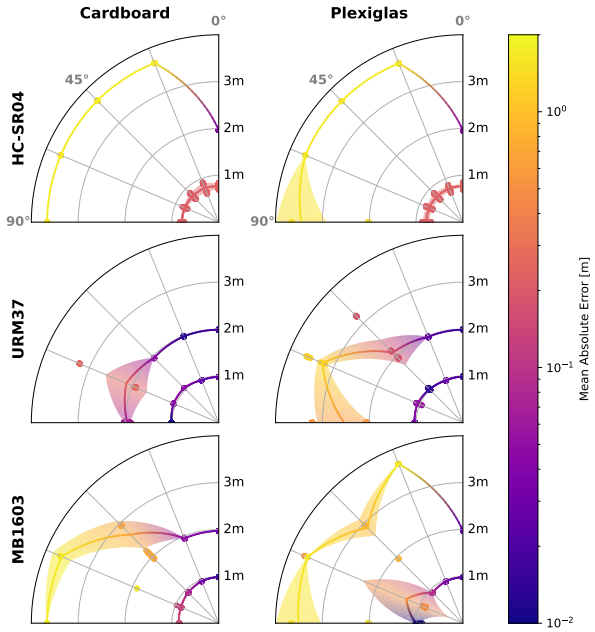
(a) Experiment 1, HC-SR04



(b) Experiment 1, URM37



(c) Experiment 1, MB1603



(d) Experiment 2 (points show measured distances)

Figure 3.3: USS experiment 1 and 2: Each distance (experiment 1: 0.25 m, 0.5 m, 1 m and 2 m; experiment 2: 1 m and 2 m) is measured 200 times for every angle (experiment 1: steps of 10°; experiment 2: steps of 22.5°). The mean value is shown by lines and linearly interpolated between the measurements. The coloured area is the standard deviation where the colour indicates the MAE between the measured and the real distance.

Chapter 4

Algorithm

4.1 Overview

VIRUS-NeRF uses *Instant-NGP* [40] as the base NeRF model. More specifically, the *Taichi implementation* [55] is employed because the original implementation is written in *Cuda* and therefore does not run on a normal computer. *Instant-NGP* is adapted in two important ways: First, *Instant-NGP* makes only use of images. Chapter 4.2 presents the addition of a depth loss to include USS and IRS measurements. Second, *Instant-NGP* uses an occupancy grid for ray-marching that is constructed through interference with the NeRF. Chapter 4.3 explains how this occupancy grid can be updated by depth measurements to improve the sampling efficiency.

No known dataset contains USS and IRS measurements. Hence, the *Robot@Home2* dataset [56] is used during development before the ETHZ dataset is collected (see chapter 5.2). It is a collection of 5 indoor apartments and more than 87k observations explored by a wheeled robot and captured by four RGB-D cameras (*Asus XTion Pro Live*). The 3D scene is reconstructed using visual odometry and an *Iterative Closest Point* (ICP) algorithm. The USS and the IRS measurements are approximated by sampling the nearest point within the FoV and a grid of 8x8 pixels from the depth images respectively.

4.2 Depth Loss

4.2.1 Depth Rendering

Instant-NGP uses only colour images for training. As shown in *iMAP* [18], the depth $\hat{D}_i$ of a pixel i can be estimated during volume rendering:

$$\hat{D}_i = \sum_{j=1}^{M} \omega_j d_j = \sum_{j=1}^{M} T_j (1 - e^{-\sigma_j \delta_j}) d_j \tag{4.1}$$

where d_j is the depth of sample j, $\delta_j = d_{j+1} - d_j$ is the distance between adjacent samples and T_j is the light transmittance (see equation 2.2). The depth rendering described in equation 4.1 is equivalent to the colour rendering of equation 2.1, except that the predicted depths d_j are summed up instead of the colours $\hat{\mathbf{c}}_j$. Analogue to the colours, the depth loss is the squared error between all estimated depths $\hat{D}_i$ and the depth measurements D_i:

$$\mathcal{L}_d = \sum_{i=1}^{N} ||\hat{D}_i - D_i||_2^2 \tag{4.2}$$

4.2.2 Depth Sensors

IRS measurements are considered to be point-like (one measurement D_i corresponds to one camera pixel) or constant in a close neighbourhood (multiple pixels have the depth D_i in a small square of pixels). The IRS depth loss $\mathcal{L}_{IRS}$ is determined directly by equation 4.2. USSs have a wide opening angle and measure the closest reflection in their FoV. Neglecting complete absorption of sound waves and specular reflections (i.e. reflections away from the sensor), two pieces of information can be drawn from each measurement D_i:

1. No obstacle in the FoV is closer than the measured distance D_i.
2. The measurement D_i is the distance to the closest obstacle in the FoV.

According to the first statement, every prediction $\hat{D}_i$ that is smaller than D_i is false and should be punished:

$$\mathcal{L}_{USS} = \sum_{i=1}^{N} ||\hat{D}_i - D_i||_2^2, \text{ for all } i \text{ where } \hat{D}_i < D_i - \epsilon_{USS} \tag{4.3}$$

$\epsilon_{USS} = 3cm$ corresponds approximately to the accuracy of the USS *URM37* (see figure 3.3b) and in this tolerance false predictions are ignored. Translating the second piece of information to a loss is more complicated because the location of the closest object is unknown. If the entire FoV is sampled during one training step, a loss between the minimal prediction and the measurement could be added:

$$\mathcal{L}_{USS\_min} = ||(\min_{\forall i \in FoV} \hat{D}_i) - D_{FoV}||_2^2 \tag{4.4}$$

However, testing shows that omitting this loss and sampling a batch of random pixels among all images is more effective. The total loss is given by the following equation:

$$\mathcal{L}_{tot} = \mathcal{L}_c + \mathcal{L}_{IRS} + \mathcal{L}_{USS} \tag{4.5}$$

where $\mathcal{L}_c$ is the colour loss from equation 2.3.

4.2.3 Rendering Bias

The depth is biased towards small values: Volume rendering is a weighted sum of distances d_j with weights $\omega_j = T_j(1 - e^{-\sigma_j \delta_j})$ (see equation 4.1). Let us assume that the densities σ_j are described by a symmetric positive function around the surface of an object (e.g. normal distribution: centre = predicted surface location, std = uncertainty). Then, the second part of the weights $(1 - e^{-\sigma_j \delta_j})$ adopts the same symmetry as the densities. The light transmittance T_j is a monotonically decreasing function (see equation 2.2). Hence, the weighting ω_j is on average larger for samples before the surface of the object than afterwards and therefore, the depth $\hat{D}_i$ is underestimated systematically. However, the NeRF is not tied to model symmetric density functions and the bias can be absorbed into the neural network. Moreover, a few samples of high density may determine the depth estimation completely because the transmittance T_j decreases exponentially and converges fast to zero.

4.3 Occupancy Grid

4.3.1 General

Instant-NGP uses an occupancy grid to skip samples during ray-marching [40]. As explained in chapter 2.4.2, the occupancy grid is updated by applying a heuristic rule and it is improved iteratively: The density predictions are utilized to refine the grid. The optimized grid enhances the efficiency of ray-marching, consequently augmenting the training process and improving the model. This, in turn, yields more accurate density predictions, thus closing the cycle of iterative improvement. *VIRUS-NeRF* updates the occupancy grid by simultaneously using the NeRF model (similar to *Instant-NGP*) and by leveraging available depth measurements. The two updating steps are called *NeRF-Update* and *Depth-Update* respectively.

4.3.2 Bayesian Update Rule

Occupancy grids strive to estimate the posterior probability $P(c_i = occ|M_1, ..., M_n)$ of a cell c_i being occupied given n measurements $M_1, ..., M_n$ [2]. Unfortunately, sensor models describe the likelihood of making a measurement M_n assuming a cell c_i is in a particular state and not vice versa. The Bayesian updating rule relates the sensor model $P(M_n|c_i = occ)$ to the posterior probability $P(c_i = occ|M_1, ..., M_n)$. Measurements and cells are considered to be independent. $P(c_i = occ)$ and $P(c_i = emp)$ are mutually exclusive, i.e. $P(c_i = occ) + P(c_i = emp) = 1$:

$$\begin{aligned} P(c_i = occ|M_1, ..., M_n) &= \frac{P(M_1, ..., M_n|c_i = occ)P(c_i = occ)}{P(M_1, ..., M_n)} \\ &= \frac{P(M_n|c_i = occ)}{P(M_n)} \frac{P(M_1, ..., M_{n-1}|c_i = occ)P(c_i = occ)}{P(M_1, ..., M_{n-1})} \\ &= \frac{P(M_n|c_i = occ)P(c_i = occ|M_1, ..., M_{n-1})}{P(M_n|c_i = occ)P(c_i = occ) + P(M_n|c_i = emp)P(c_i = emp)} \end{aligned} \tag{4.6}$$

where $P(M_n)$ is the prior probability of taking the measurement M_n and $P(c_i = occ|M_1, ..., M_{n-1})$ is the occupancy grid before integrating M_n.

The *Depth-Update* and the *NeRF-Update* use both the Bayesian updating rule from equation 4.6. The *Depth-Update* utilize the USS and IRS sensor models described in chapter 4.3.3. The *NeRF-Update* uses no conventional sensor model. However, the term $P(M_n = \sigma_i|c_i = occ)$ can be thought of as the probability of the NeRF predicting or "measuring" the density σ_i given that a cell is occupied.

4.3.3 Depth-Update

The *MURIEL* method [21] is the best real-time occupancy grid implementation for USS [23] (see chapter 2.2). Therefore, it is used as the sensor model for USSs and IRSs in this work. The *Multiple Target Model* of the *MURIEL* method calculates the probability that a measurement result M_n is made at distance D assuming a cell c_i is either occupied ($c_i = occ$) or empty ($c_i = emp$). Equivalently, one can write that $M_n = D$ and no reading M_n is smaller than D, i.e.:

$$P(M_n|c_i) = P(M_n = D|c_i)P(M_n \not< D|c_i) \tag{4.7}$$

Every measurement has a certain probability of falsely detecting a target with constant probability P_F:

$$P(M_n = D|c_i = emp) = P_F \tag{4.8}$$

The sensor measurements are considered to be Gaussian using a standard deviation of $\sigma(M_n) = M_n\sigma_{every-meter}$ where $\sigma_{every-meter}$ is a hyper-parameter:

$$P(M_n = D|c_i = occ) = e^{-\frac{(D-M_n)^2}{2\sigma(M_n)^2}} + P_F \tag{4.9}$$

Adding P_F implies that there is a small probability that another object than the target triggers the measurement. The probability that the measurement is not smaller than distance D is given for an occupied ($c_i = occ$) or empty ($c_i = emp$) cell by:

$$P(M_n \not< D|c_i) = 1 - \int_0^D P(M_n = x|c_i)dx \tag{4.10}$$

Finally, the above equations can determine the measurement probabilities $P(M_n|c_i = occ)$ and $P(M_n|c_i = emp)$ which are used to update the occupancy grid (see equation 4.6). The presented *Multiple Target Model* is extended to a *Cone Model* to represent the cone-shaped propagation of sound waves [21]. Reflecting properties are modelled by the *Specular Model*. These two extensions are specific to USSs and are not applied to IRSs. The *Depth-Update* is implemented using USSs and IRSs together and both individually. Testing on the *Robot@Home2* dataset [56] shows that using USSs worsens the results. This might be due to the low angular resolution of USS preventing the occupancy grid from becoming finer. Therefore, the *Depth-Update* is done uniquely with IRSs.

4.3.4 NeRF-Update

Before starting the learning process, the *Taichi implementation* of *Instant-NGP* excludes cells that are out of sight making the occupancy grid update more efficient. *VIRUS-NeRF* cannot assume that all measurements are available at the beginning of the training. Therefore, the data is sampled in the direction of USS depth measurements to focus on relevant regions. Some noise is added to the positions of the samples to have a higher variability.

The MLP of *Instant-NGP* outputs density predictions between zero and infinity. *VIRUS-NeRF* calculates the probability of a density prediction by projecting the density from $[0, \infty)$ to $[0, 1]$:

$$P(M_n = \sigma_i|c_i = occ) = \frac{1}{1 + (\frac{\sigma_T}{\sigma_i})^\zeta} \tag{4.11}$$

where σ_i is the predicted density by the NeRF, ζ is the slope of the mapping function and σ_T is the density threshold. If $\zeta \longrightarrow \infty$, then the projection $P(M_n = \sigma_i|c_i = occ)$ becomes a step function. If $\sigma_i > \sigma_T$, then the occupancy probability $P(M_n = \sigma_i|c_i = occ)$ is larger than 0.5 and vice versa. The weights of the density MLP are randomly initialized in $[-\frac{1}{\sqrt{32}}, \frac{1}{\sqrt{32}}]$ leading to small σ_i values in the beginning of the training. If σ_T is fixed, then $P(M_n = \sigma_i|c_i = occ)$ would vanish in the first few cycles. Therefore, σ_T is defined as a function of σ_i as follows:

$$\sigma_T = \min(\sigma_{Tmax}, \frac{1}{N}\sum_{i=1}^{N}\sigma_i) \tag{4.12}$$

where σ_{Tmax} is the maximum density threshold and N is the batch size. For all tested σ_{Tmax}, equation 4.12 becomes constant after few training steps: $\sigma_T = \sigma_{Tmax}$.

Chapter 5

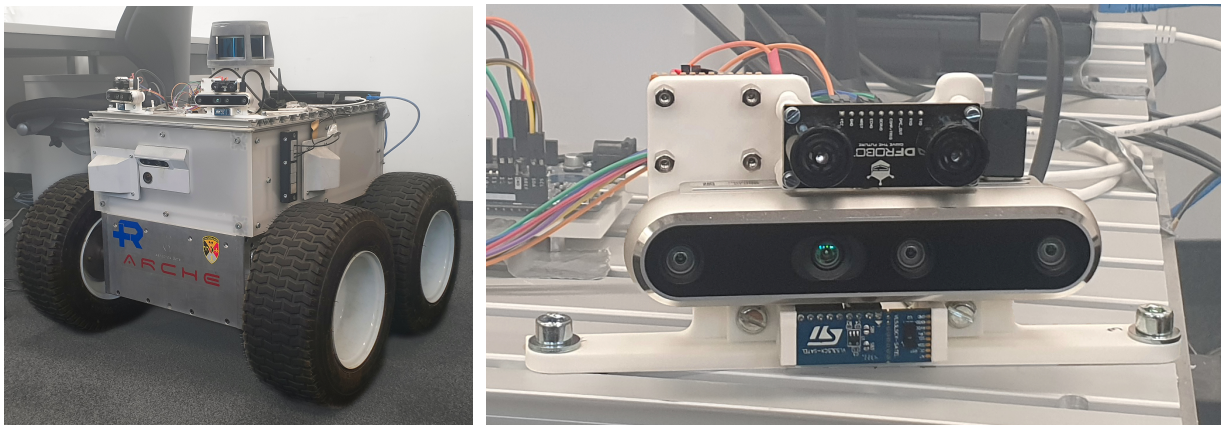
Mapping Experiments

5.1 Overview

In this chapter, *VIRUS-NeRF* is evaluated on a real-world dataset. No public dataset exists containing USS, IRS and camera measurements. Therefore, a custom one is created at ETHZ. The sensor setup and the data collection are explained in chapter 5.2. Afterwards, the evaluation metrics are introduced in chapter 5.3. Although the NeRF algorithm assumes that the poses are known, they have to be estimated for the dataset. In chapter 5.4.1, bundle adjustment is used to improve the pose estimation. Further, the hyper-parameters of *VIRUS-NeRF* are optimized by using *Particle Swarm Optimization* (PSO) in chapter 5.4.2. Chapter 5.5 discusses the local mapping results of multiple scenes including an exhaustive ablation study which explains the contribution of the different sensors, the algorithm parts and the optimization techniques. During these tests, the training is done in an offline fashion where all data is available already at the beginning. However, in an online robotic system, *VIRUS-NeRF* would be trained simultaneously while exploring the environment. Hence, chapter 5.6 compares the convergence of offline and online training.

5.2 Dataset

5.2.1 Setup



(a) Super Mega Bot

(b) Sensor stack: USS, camera and IRS (top to bottom)

Figure 5.1: Experimental setup

The *Super Mega Bot* (SMB) is a differential drive robot designed by *Inspector Bots* [57] (see figure 5.1a). On top of the SMB, a *RS-LiDAR 16* and two sensor stacks are fixed. The sensor stack mounts the USS, the IRS and the *RealSense D455* camera (see figure 5.1b). The sensor stack is designed such that all sensors of one stack are oriented in the same direction and are as close as possible.

The experiments are logged on a *Ubuntu 20.04* server running *ROS Noetic*. The cameras are connected directly to the computer via USB-C cables. Using PWM and I2C, the USSs and IRSs respectively communicate their measurements to an *Arduino Zero* which transmits them to the computer via *Rosserial*. To reduce the number of jumper cables, two types of PCBs are soldered to collect and redistribute common signals, i.e. ground, 5 V, 3.3 V, SDA and SCL. One of those PCBs is fixed close to the sensors on the backside of the sensor stack and the other one is beside the Arduino. The *RealSense D455* offers hardware synchronization and triggers the USS directly. The triggering signal is active low at 1.8 V. The *URM37* requires a signal which is active high at 3.3 V. Therefore, a transistor is used to invert the signal and convert it from 1.8 V to 3.3 V.

The number of sensors is limited by the availability of three *RealSense D455* cameras. Initially, three sensor stacks are created pointing to the left, right and front of the robot. This is expected to be a good configuration because NeRFs benefit from a variety of viewing directions [37]. Unfortunately, the USB-C connection of one of the *RealSense D455* cameras broke before conducting the experiments and could not be replaced. Therefore, the experiments are made with two sensor stacks pointing to the front left and front right of the SMB. The sensor stacks are approximately 27 cm apart at $\pm 14.5°$ to the driving direction.

5.2.2 Sensor Synchronization

VIRUS-NeRF assumes that the measurements of one stack are made simultaneously. Therefore, all three sensors should be synchronized. However, the IRS does not provide any functionality to be triggered. Hence, it is recorded at a higher rate (15Hz) than the other sensors (5Hz) and the closest sample in time is selected during post-processing (see figure C.1). To reduce the error, the robot is moved slowly during the experiments. As described in chapter 5.2.1, the USS is triggered directly by the *RealSense* at 5Hz. Unfortunately, the logging rate of the *RealSense* unveils to be only 4.81Hz after making the experiments while the USS has a frequency of 5Hz. Because the USS has the correct measurement rate, the camera is expected to operate at 5Hz and the signal frequency is reduced during transmittance. Similar to the IRS, the closest USS sample in time is assigned to each image as shown in figure C.1.

5.2.3 Sensor Calibration

By localizing a target board with known dimensions, the calibration tool *Kalibr* [58], [59], [60], [61] estimates the intrinsic and extrinsic camera parameters simultaneously. *Kalibr* requires the cameras to have an overlapping FoV. This is why the sensor stacks are oriented at $\pm 14.5°$ to the driving direction. The recommended calibration by recording a Rosbag while moving around the calibration target does not work because the cameras are not synchronized. Therefore, the calibration target is attached to a microphone stand (see figure C.3) and one image is taken at a time. The camera calibration results can be found in chapter C.1.2.

The LiDAR is considered to be at the horizontal centre of the robot. To determine the relative pose of the cameras to the LiDAR, the *Camera-LiDAR Calibration - V 2.0* [62] is used. Similar to *Kalibr*, the camera estimates the position of a calibration target by detecting a chessboard-like structure (see figure C.3). The LiDAR determines the pose of the target by fitting a plane on the surface of the board. Getting a reasonably

low calibration error is a tricky process because the point cloud of the *RSLiDAR 16* is not very dense which makes the plane fitting difficult. The camera-to-LiDAR calibration results are shown in figure C.4.

The USS and the IRS are not calibrated to the camera. As discussed in chapter B.1, USS have a low angular resolution. In addition, the USS experiments show that the FoV of the sensor depends on the material and orientation of the target (see chapter 3.3.2). Therefore, the translation and rotation between the USS and the camera are negligible. The same argument cannot be made for the IRS because it has a much higher angular resolution. Hence, the behaviour of the error is analysed on the *Robot@Home2* dataset [56] (dataset used before the one at ETHZ is collected, see chapter 4.1): An artificial orientation error is added to the simulated IRS data. Training multiple models with increasing errors shows that the mean *Nearest Neighbour Distance* (NND) is not influenced by an angular error up to 3° (see figure C.5). In this test, the IRS is simulated to have a beam size of one pixel. The effective beam size is expected to be larger, leading to a smaller influence of an angular error. 3° are in the range of the mechanical precision and therefore, the IRS is not calibrated to the camera.

5.2.4 Data Processing

The measurements are recorded in *Rosbags* and the dataset is created as follows:

- Measurements: Crop *Rosbags* in time. Reduce the FoV of the LiDAR to filter dynamical objects and synchronize USS, IRS and camera measurements (see chapter 5.2.2).
- Poses: Estimate poses with *KISS-ICP* [63]. Optimize poses with *BALM* [64] (see chapter 5.4.1) and linearly interpolate them to be synchronous with the camera measurements.
- GT map: Project the LiDAR point clouds to a grid in world coordinates having a $3\,\mathrm{cm}^3$ cube size and by using the optimized *BALM* poses. Threshold the grid at a minimum of two points per cell to reduce noise.
- Train-Test split: 80% of all measurements are used for training, 10% are used for testing and the remaining 10% could be used for evaluation.

5.2.5 Experiment Environment

The experiments are conducted at ETHZ at the following places:

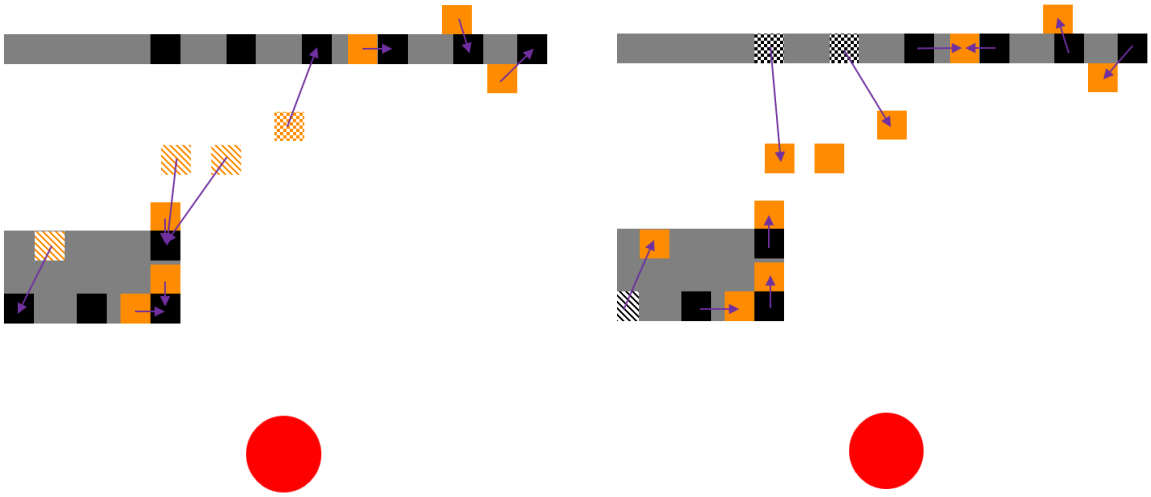
- *Office*: office room of $72\,\mathrm{m}^2$ (LEE building, ASL)
- *Common-Area*: room with tables, couches and a kitchen corner of $216\,\mathrm{m}^2$ (LEE building, ASL)
- *Corridor*: long narrow corridor of $240\,\mathrm{m}^2$ (CLA building)

Most environments are recorded multiple times until a quasi-static scene is captured where only small movements are visible, e.g. a person writing on a keyboard. Some experiments are made in the garage of the LEE building. Unfortunately, the pose estimation of *KISS-ICP* [63] breaks when rotating on the trajectory. The garage is a much larger room. Therefore, the measurement points are further away having a reduced accuracy and an increased variation when rotating. The ICP algorithm seems to be affected by these two factors. Undistorting the point cloud could fix this problem.

5.3 Evaluation Metrics

Collision avoidance, obstacle circumnavigation and other local mapping applications are traditionally done using instantaneous depth measurements. Therefore, *VIRUS-NeRF* is compared to momentary scans of USSs, IRSs and LiDARs. For each test point, an error is calculated between the predicted local map and the real one. For the sensors, the prediction is the instantaneous depth measurement and for *VIRUS-NeRF*, it is a 360° depth scan obtained by volume rendering (see equation 4.1). The *Ground Truth* (GT) consists of a 360° depth scan inside the global map which is based on LiDAR point clouds (see chapter 5.2.4). The sensors and *VIRUS-NeRF* are not directly assessed against the global map to mitigate the erroneous association of depth predictions to incorrect objects within the global map. The 2D scans of *VIRUS-NerF* and the GT are performed at the height of the camera. The 3D depth measurements are collapsed to a 2D representation in a vertical range of ±5 cm above and below the camera height.

A simple metric would be the RMSE between the GT scans and the depth predictions. However, the RMSE is sensible to small orientation errors which becomes obvious when considering a close object in front of a distant background. In this case, a small angular error leads to a very large RMSE at the edges of the object. In consequence, the *Nearest Neighbour Distance* (NND) is used for evaluation. The NND can be calculated in two directions: The NND from the $Sensor \rightarrow GT$ calculates the Euclidean distance from a prediction to the closest point in the GT (see figure 5.2a). This metric describes the accuracy of the prediction. The inverse NND from the $GT \rightarrow Sensor$ is the distance from a GT point to the closest prediction which is a measure of coverage (see figure 5.2b). The USS and the IRS have by design limited FoVs. In this project, the FoV of the LiDAR is restricted to filtering dynamic objects. Limiting the GT to the FoV ($GT[FoV] \rightarrow Sensor$) is distinguished from the coverage all around the robot ($GT[360°] \rightarrow Sensor$).



(a) Accuracy: $Sensor \rightarrow GT$ (60% inliers, 10% outliers *too close*, 30% outliers *too far*)

(b) Coverage: $GT \rightarrow Sensor$ (70% inliers, 20% outliers *too close*, 10% outliers *too far*)

Figure 5.2: Exemplary metrics: NND as violet arrows, robot in red, GT map in grey, GT scan in black and sensor/NeRF prediction in orange. Outliers: squared pattern means that prediction is *too close* (closer to the robot than GT), diagonal lines indicate them to be *too far*.

Three quantities are calculated over all test points: The mean NND, the median NND and the percentage of inlier points. Inliers are defined as points where the NND is less than 10 cm. The outliers can be separated into two groups: First, outliers where the GT depth is larger than the predicted one. These are outliers that are *too close*. Second, there exist outliers that are *too far* away. Figure 5.2 shows that the ratio between *too close* and *too far* outliers depends on the direction of the NND. This project focuses on the outlier statistic of the coverage ($GT \rightarrow Sensor$) because it declares outliers based on the GT which is easier to interpret than outliers based on predictions. The assessment of accuracy, coverage and inlier ratio aligns with the evaluation criteria employed in *iMap* [18] and *Nice-SLAM* [19], wherein the mapping coverage is denoted as *completion* and the proportion of inliers as *completion ratio*, using a threshold of 5 cm instead of 10 cm.

All metrics are determined for three zones defined by the GT depth. The zones roughly represent different applications: The first zone ($0-1$ m) concerns safety applications, e.g. collision avoidance. The second zone ($0-2$ m) is relevant for local tasks, e.g. obstacle circumnavigation. The third zone ($0-100$ m) is ideal for global path planning.

5.4 Optimization

5.4.1 Bundle Adjustment

KISS-ICP is an odometry pipeline based on LiDAR point clouds [63]. It uses point-to-point ICP to estimate the pose of the robot. For long trajectories, the poses may drift due to the accumulation of small errors. The drift becomes more important for large environments in which the entire scene is not observable at once. NeRFs depend on an accurate pose estimation. Therefore, the poses are corrected by *Bundle Adjustment for LiDAR Mapping* (*BALM*) determining planes for point association [64]. *BALM* requires an initial pose estimation given by *KISS-ICP*. The ablation study demonstrates how pose optimization improves significantly the accuracy, especially for large scenes (see chapter 5.5.1). The odometry could be enhanced by integrating an IMU or using wheel odometry. Presumably, the LiDAR-to-camera calibration becomes quickly the limiting factor.

5.4.2 Particle Swarm Optimization

Motivation

VIRUS-NeRF has lots of hyper-parameters including parameters for the training and the occupancy grid. In total, at least 14 parameters are essential. Standard grid search with a mesh of 5 steps per parameter results in more than $6 * 10^9$ configurations. Considering a training time of 45 seconds per model, this problem is intractable.

Working Principle

Particle Swarm Optimization (PSO) is a stochastic optimization technique [65]. Inspired by schools of insects, birds and fishes, it is based on the behaviour of individual particles inside a swarm. The particles travel through the hyper-parameter space with a certain velocity and try to find the optimal set of parameters. At each PSO iteration, the velocity and position of every particle are updated by the following equations:

$$\mathbf{v}_i = \alpha_m \mathbf{v}_{i-1} + \alpha_p r_{i,p}(\mathbf{x}_{i-1}^{best} - \mathbf{x}_{i-1}) + \alpha_s r_{i,s}(\mathbf{x}_{i-1}^{nbh} - \mathbf{x}_{i-1}) \tag{5.1}$$

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \mathbf{v}_i \tag{5.2}$$

where $\mathbf{x}_{i-1}^{best}$ is the best previous position of the particle, $\mathbf{x}_{i-1}^{nbh}$ is the best position in the neighbourhood of the particle and $r_{i,p/s}$ is a random number drawn from $[0, 1]$ at

iteration i. α_m, α_p and α_s are constant hyper-parameters weighting the momentum, the proper and the social behaviour respectively. Then, the current position $\mathbf{x}_i$ is evaluated and $\mathbf{x}_i^{best}$ and $\mathbf{x}_i^{nbh}$ are updated.

The complexity of PSO is $\mathcal{O}(\#iterations * \#particles)$ and does not depend on the number of hyper-parameters. Instead of testing a new set of parameters at every iteration, the best one is re-evaluated from time to time. This trade-off between exploration and exploitation is necessary because otherwise, the algorithm could converge to a sub-optimal solution due to a "lucky" one-time evaluation. The velocity of the particle is reflected at the hyper-parameter border to prevent an accumulation of particles in these regions.

Optimization

For testing and setting the PSO hyper-parameters, the algorithm is simulated with two parameters on an artificial metric shown in figure C.6. If not mentioned otherwise, the number of particles is equal to 32, the neighbourhood is defined by the 5 closest particles (including the proper one), the exploration and exploitation probabilities are 75% and 25% respectively and $\alpha_m = 0.65$, $\alpha_p = 0.25$ and $\alpha_s = 0.25$. When optimizing the real dataset, the mean NND of zone 3 ($0 - 100\,\mathrm{m}$) is used as the evaluation metric. Each model is trained for 500 steps or a maximum of 45 seconds.

Figure 5.3 shows the optimization over 48 iterations corresponding to approximately 18h runtime on a *Nvidia Titan Xp* GPU. The first row of the figure 5.3 indicates that the particles converge already after about 25 iterations. The second row shows the NND statistics for each particle over the last 10 iterations. From the box plots, the five best particles with the lowest NND can be determined. In the third row, the converged values are plotted for the best 5 particles. In each neighbourhood, the particles tend to converge to the same solution. For example, the best three particles (2, 30 and 31) and the fourth and fifth (1 and 9) converge to two different sets of parameters. This shows the ability of PSO to produce a variety of candidate solutions. The two best neighbourhoods (particles 2-30-31 and 1-9) have some common properties: The IRS loss (*tof loss w*) converges to a high value while the USS and the colour losses are much lower. This proposes that the IRS is particularly important during training. The batch ratio of the occupancy grid (*batch ratio ray update*) is neither very high nor very low suggesting that the dual updating technique of the occupancy grid is advantageous compared to using exclusive the *NeRF-Update* or the *Depth-Update* (see chapter 4.3). In contrast to the USS, no extensive tests are made with the IRS because it is added at a later stage in the project. Hence, the beam size of the IRS (*tof pix size*) is optimized as hyper-parameters. The particles do not converge to a distinct value but suggest that the beam size is at least 7 camera pixels large.

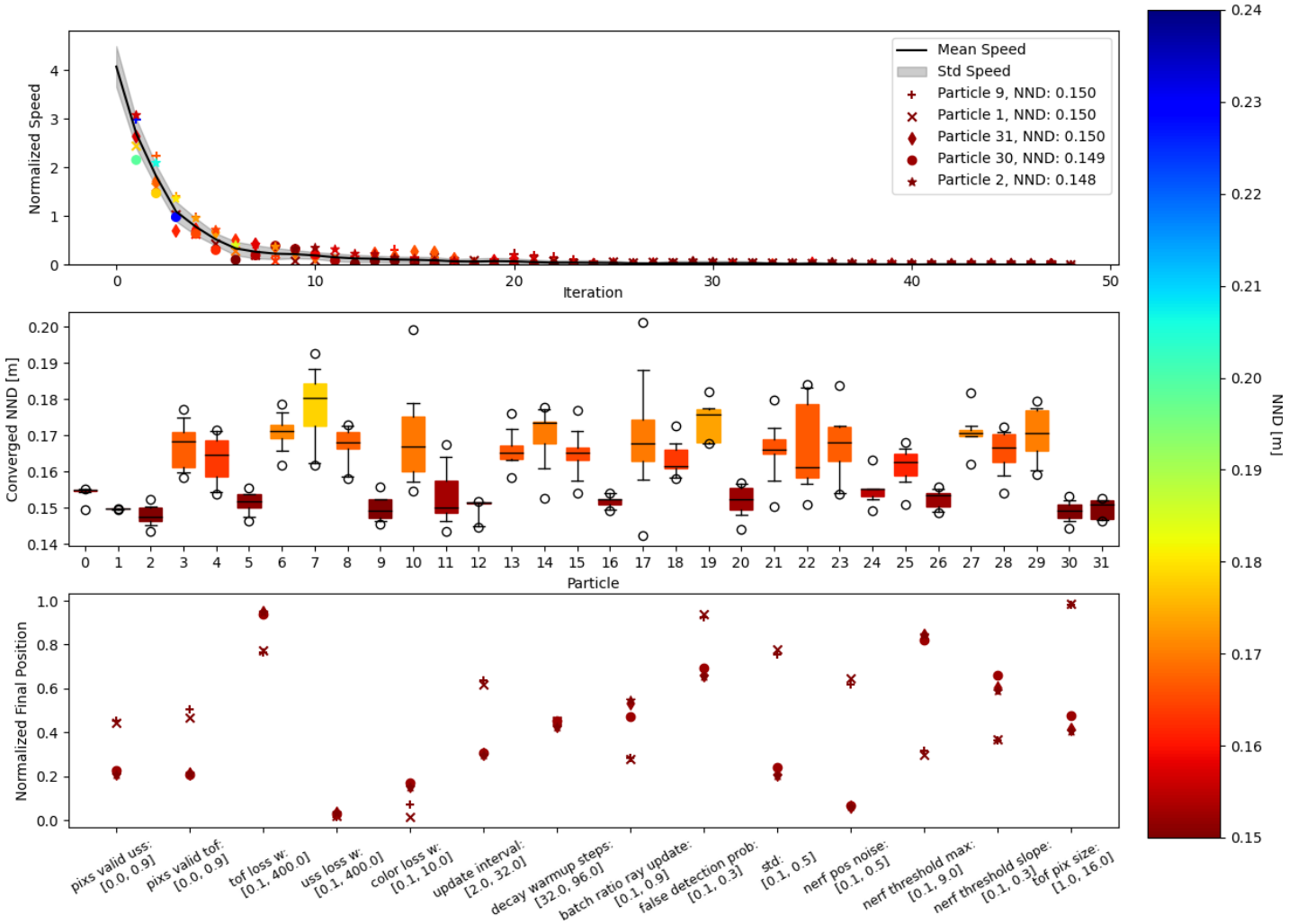


Figure 5.3: PSO: Colours indicate the NND in meters (*Sensor* → *GT*, zone 3). The first row shows the normalized mean speed over all particles and the values of the five best ones. The second row shows box plots of the last 10 iterations: The boxes expand [25%,75%] and the bars [10%,90%] of the values. The black lines mark the mean and the circles are outliers. The third row shows the normalized final parameters of the best five particles (particles 2, 30, 31, 1 and 9). The non-normalised search space is indicated below the parameter names.

5.5 Local Mapping

5.5.1 Results

Nearest Neighbour Distance

The best hyper-parameters are used for each room (*office*: optimized parameters of particle 2; *common area*/*corridor*: hand-tuned parameters, see later in this chapter). Each model is trained for 800 steps or a maximum of 80 seconds. For the *office* and the *common area*, two sample maps are shown in the figures 5.4 and 5.5 respectively. Further results are found in chapter C.3.1 including the *corridor* and using RGB-D cameras.

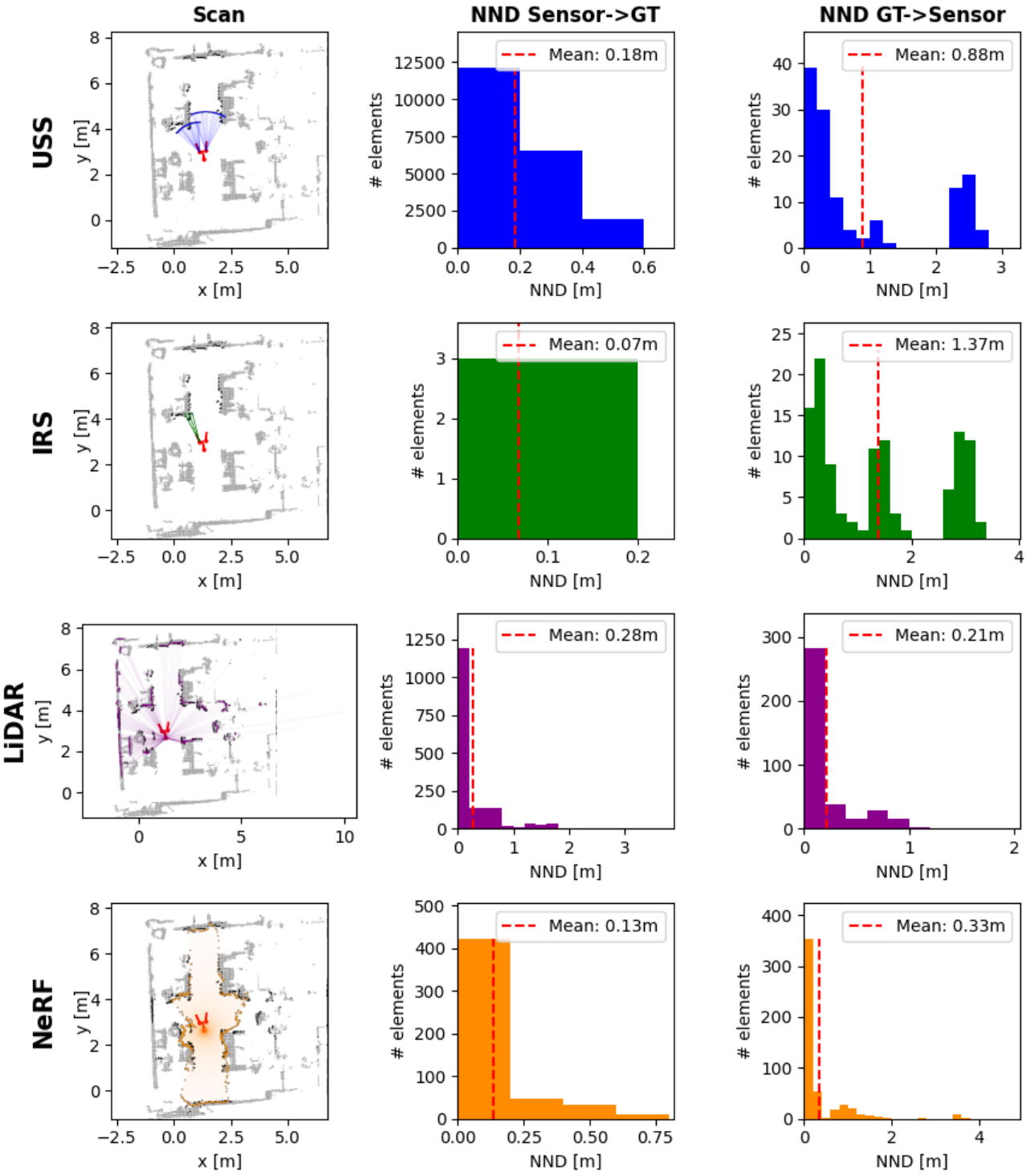


Figure 5.4: *Office* map (test point 14): In the first row, the global map is in grey and the GT scan is in black. Columns 2 and 3 show histograms of the number of measurements as a function of the NND for the accuracy and coverage respectively. The bin size is 0.2 m. The GT is limited to the FoV ($GT[FoV] \rightarrow Sensor$).

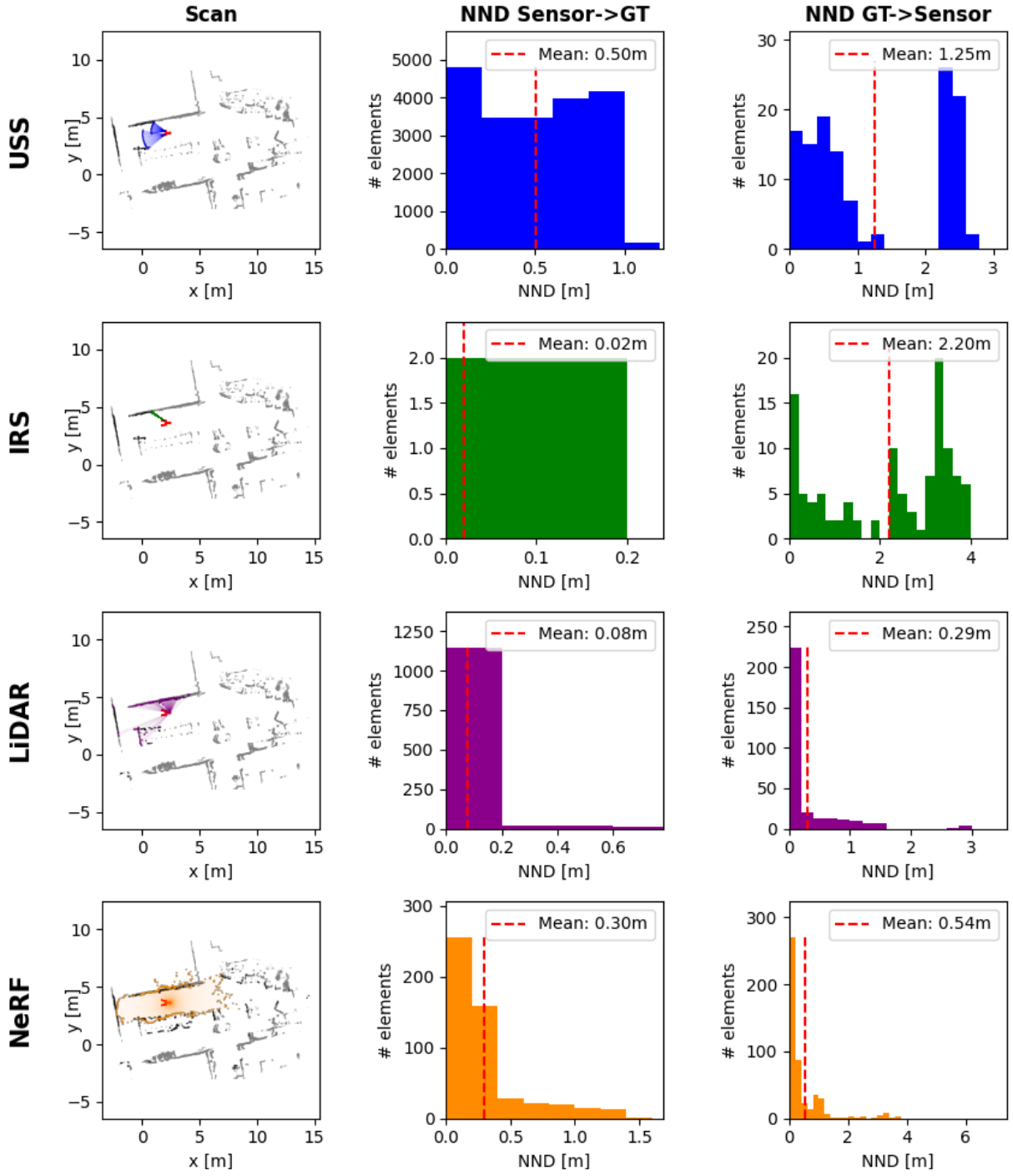
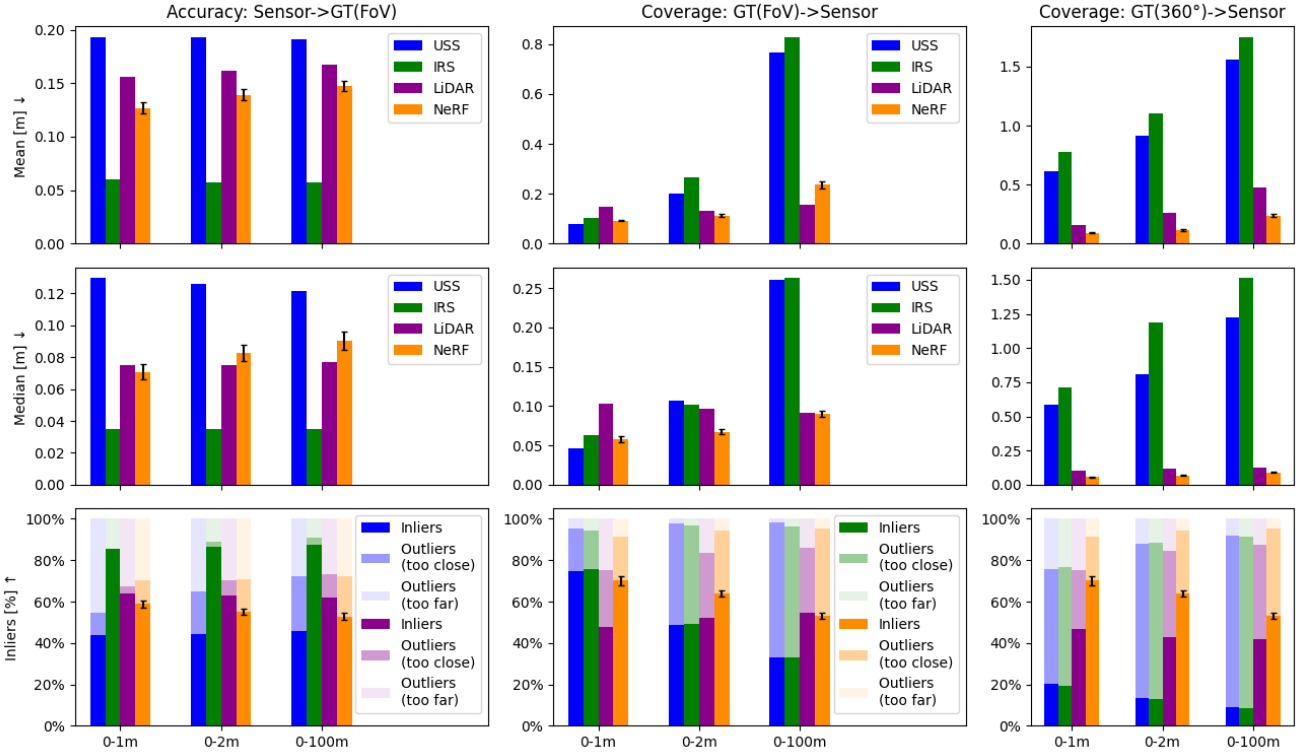
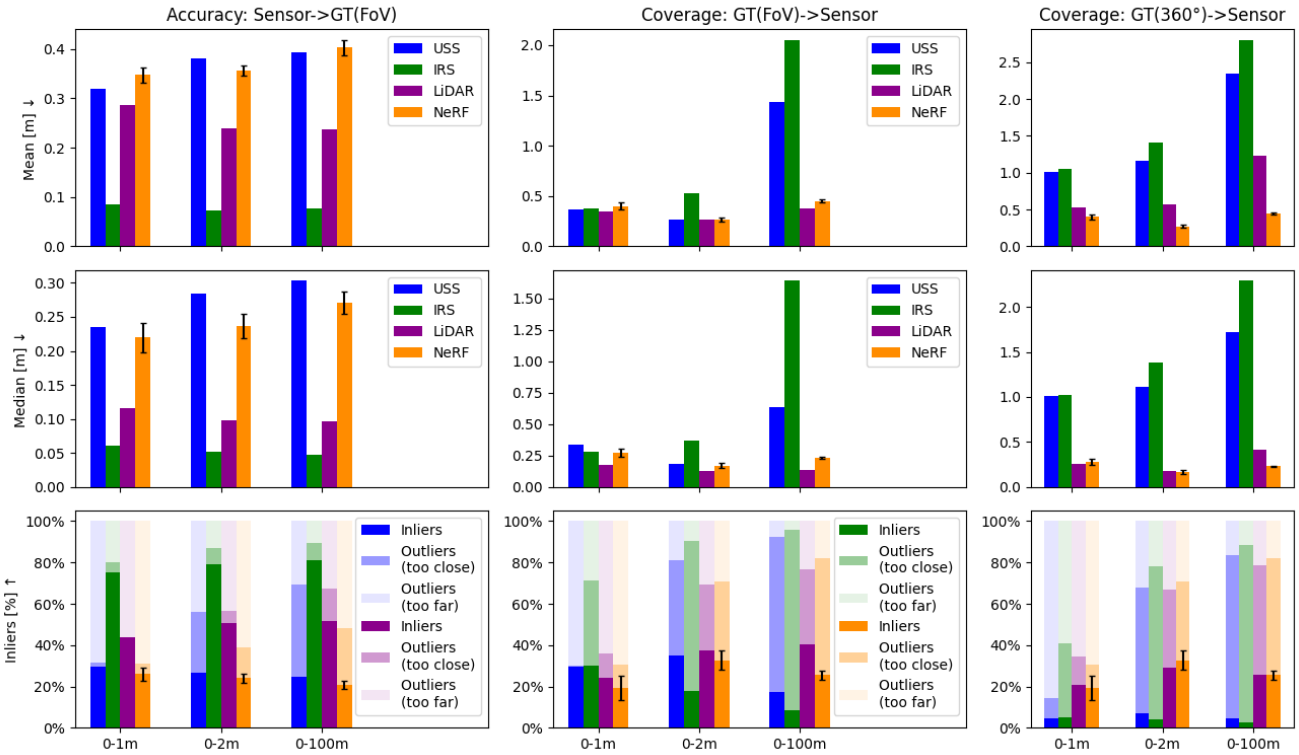


Figure 5.5: *Common area* map (test point 35): In the first row, the global map is in grey and the GT scan is in black. Columns 2 and 3 show histograms of the number of measurements as a function of the NND for the accuracy and coverage respectively. The bin size is 0.2 m. The GT is limited to the FoV ($GT[FoV] \rightarrow Sensor$).

The average statistics for all test points and over 10 runs are summarised in figure 5.6 (see figure C.12 for the *corridor*). While the amplitude of the metric depends on the particular environment, the tendency is everywhere likewise: The USS has neither good accuracy nor coverage. The IRS achieves the best accuracy while having the worst coverage. LiDARs retain an accuracy between IRSs and USSs while having the best coverage of all sensors, especially in the third zone ($0 - 100$ m). Depending on if the GT is limited to the FoV ($GT[FoV] \rightarrow Sensor$) or not ($GT[360°] \rightarrow Sensor$), *VIRUS-NeRF* scores slightly worse respectively better than the coverage of the LiDAR. The accuracy of the NeRF depends on the scene: For the *office* it is comparable to the LiDAR but for the *common area* it exhibits performance akin to USSs.



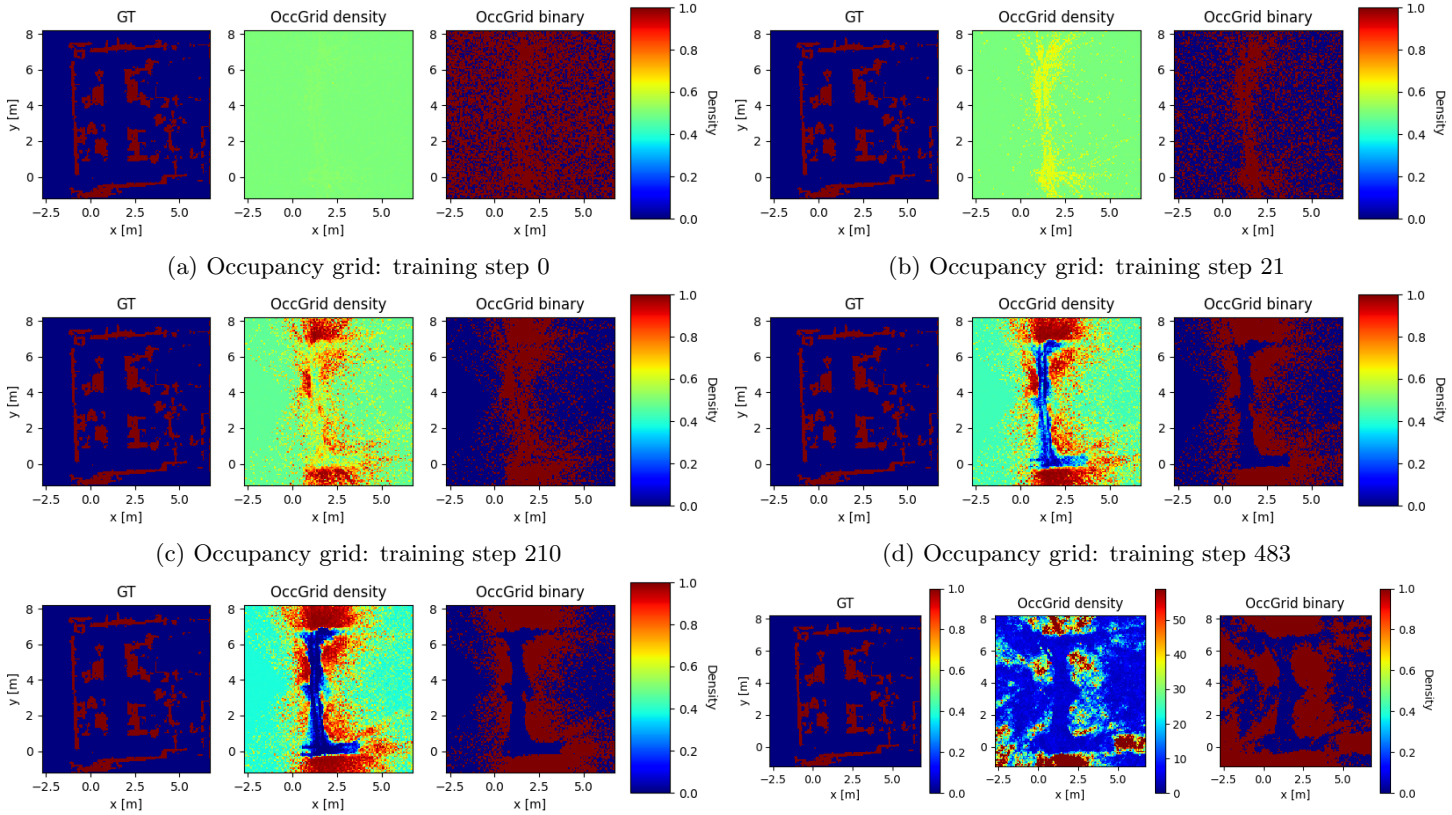
(a) *Office*



(b) *Common area*

Figure 5.6: NND results: The first column describes the accuracy ($Sensor \rightarrow GT$) and the second and third ones the coverage limited to the FoV ($GT[FoV] \rightarrow Sensor$) or using all-around GT ($GT[360°] \rightarrow Sensor$). The rows show the mean NND, the median NND and the inlier ($NND < 10cm$) percentage. Outlier predictions are either *too close* or *too far* away relative to the robot. Each metric is calculated for three zones defined by the GT depth. *VIRUS-NeRF* is evaluated for 10 runs and the error bar indicates the standard deviation.

Occupancy Grids



(a) Occupancy grid: training step 0

(b) Occupancy grid: training step 21

(c) Occupancy grid: training step 210

(d) Occupancy grid: training step 483

(e) Occupancy grid: training step 798 (final)

(f) *Instant-NGP* grid: training step 480 (final)

Figure 5.7: *Office* occupancy grids for several training steps: The grids are generated by *VIRUS-NeRF* except for subplot f which shows the final *Instant-NGP* grid.

Figure 5.7 shows the occupancy grid for different training steps for the *office*. For the *common area*, similar results can be found in figure C.13. The occupancy grid is initialized randomly slightly above the threshold of 0.5 (see figure 5.7a). During some warm-up steps, the density is reduced exponentially to clear the free space (see figure 5.7b). The final result resembles the grid of *Instant-NGP* [40] (compare figures 5.7e and 5.7f).

Ablation Study

The PSO optimization is conducted with the data from the *office* environment. For the ablation study, the models are evaluated on the *common area* to see if the optimized parameters generalize to other environments. The mean, median and inlier percentage are shown in table 5.1. The RGB-D camera outperforms all other sensor constellations. When removing either the USS or the IRS, the performance drops. The second-best results are achieved by using the optimised poses and the hand-tuned hyper-parameters (not optimized by PSO). The original *Instant-NGP* implementation is significantly worse in all metrics. When adding the depth losses and still using the occupancy grid of *Instant-NGP*, the results improve but do not reach the same level as *VIRUS-NeRF*. The same ablation study is repeated for the *office* environment (see table C.1). The results are similar except that the optimized hyper-parameters perform better than the hand-tuned ones. Also, omitting the pose optimization is less severe than in the *common area*.

		Mean [m] ↓		Median [m] ↓		Inliers [%] ↑	
		Sensor →GT	GT(360°) →Sensor	Sensor →GT	GT(360°) →Sensor	Sensor →GT	GT(360°) →Sensor
Camera	mean	0.704	1.412	0.627	1.051	0.052	0.056
	std	*0.056*	*0.680*	*0.102*	*0.476*	*0.037*	*0.041*
Camera & USS	mean	0.49	0.568	0.358	0.381	0.146	0.146
	std	*0.095*	*0.028*	*0.083*	*0.074*	*0.046*	*0.038*
Camera & IRS	mean	0.625	0.643	0.569	0.392	0.097	0.169
	std	*0.026*	*0.131*	*0.028*	*0.107*	*0.020*	*0.034*
RGB-D	mean	0.324	0.378	0.171	0.134	0.324	0.389
	std	*0.008*	*0.011*	*0.006*	*0.003*	*0.012*	*0.012*
NGP	mean	0.712	1.287	0.623	0.992	0.056	0.059
	std	*0.071*	*0.584*	*0.091*	*0.403*	*0.030*	*0.032*
NGP Grid	mean	0.509	0.501	0.402	0.304	0.131	0.22
	std	*0.009*	*0.011*	*0.006*	*0.013*	*0.004*	*0.008*
Poses not optimized	mean	0.728	0.922	0.535	0.377	0.113	0.186
	std	*0.014*	*0.072*	*0.034*	*0.016*	*0.017*	*0.012*
Params not optimized	mean	0.403	0.448	0.27	0.23	0.206	0.256
	std	*0.016*	*0.014*	*0.017*	*0.009*	*0.020*	*0.022*
Optimized Particle 1	mean	0.43	0.531	0.305	0.271	0.198	0.277
	std	*0.058*	*0.350*	*0.087*	*0.211*	*0.078*	*0.088*
Optimized Particle 2	mean	0.462	0.52	0.353	0.275	0.152	0.226
	std	*0.035*	*0.134*	*0.033*	*0.031*	*0.014*	*0.028*

Table 5.1: Ablation Study *Common Area*: The NND is calculated for zone 3 ($0-100\,\mathrm{m}$) and averaged over 10 runs showing the standard deviation. All results are generally produced by *VIRUS-NeRF* with the optimized poses and not-optimized hyper-parameters (third last row). The first four rows show different sensor constellations by removing USSs and IRSs or by using depth cameras. The fifth row is the unchanged *Instant-NGP* implementation using only cameras and the original occupancy grid. In row 6, the USS and IRS losses are added to *Instant-NGP*. In rows 7 and 8, the poses or the hyper-parameters are not optimized respectively. The last two rows use optimized parameters from PSO particles 1 and 2. Dark orange, light orange and yellow indicate the best, second and third-best results respectively.

5.5.2 Discussion

Nearest Neighbour Distance

The results reflect the theoretical advantages and weaknesses of the sensors (see table B.1): The accuracy of the USS is limited by the poor angular resolution. Up to zone 2 ($0-2\,\mathrm{m}$), the coverage of the USS within the FoV is close to the one of the LiDAR due to its large opening angle. However, in the third zone ($0-100\,\mathrm{m}$), the coverage of the USS worsens significantly. These observations are in line with common USS applications used for short-range tasks where coverage is more important than accuracy, e.g. for car parking assistants [66]. The IRS measurements are very sparse. Therefore, it misses some objects completely which is visible on the histograms of figures 5.4 and 5.5 leading to the worst coverage of all sensors. The accuracy of the IRS exceeds all other sensors even though the IRS is a rather cheap sensor. The higher range of the LiDAR does not explain the better accuracy of the IRS because it is also present in zones 1 and 2. However, the LiDAR sensor is a better trade-off between accuracy and coverage than the IRS. The map in figure 5.4 reveals that sometimes the narrow beam of the LiDAR passes through holey areas that are considered to be occupied in the GT. Whereas

the point clouds contain enough correct measurements to avoid collisions, they require more filtering/post-processing compared to other sensors. For all sensors, outliers tend towards being too close to the robot (see coverage in figure 5.6). Measurement beams prolongate at most up to the surface of the object when neglecting opaque materials and specular reflections away from the sensor. This may explain why outliers of sensors tend to be closer to the robot than further away.

How the accuracy of *VIRUS-NeRF* depends on the optimization is further discussed in chapter 5.5.2. The coverage is equivalent to LiDAR scans. Looking at the maps of figures 5.4. 5.5, C.7 and C.9, the NeRF algorithm detects almost all objects. If the NeRF algorithm partially overlooked some structures, this would be visible by high NND peaks inside the $GT[FoV] \rightarrow Sensor$ histograms like for the USS and the IRS. The maps show that *VIRUS-NeRF* tends to underestimate the distance to objects leading to false-positive predictions. As explained in chapter 4.2.3, the volume rendering is biased towards underestimating the depth. If this bias is not fully absorbed by the neural network, it may offset the depth predictions. This phenomenon is visible in the outlier statistics of all three environments (see figures 5.6 and C.12): Considering outliers related to the coverage, there are more predictions *too close* (closer to the robot than the GT) than *too far*. For accuracy, this trend is inverse but less meaningful as explained in chapter 5.3. Most of the false-positive predictions are further away from the current position of the robot and the hallucinated objects disappear if the robot moves towards them (e.g. compare points at $x = 2.5\,\text{m}$, $y = 0\,\text{m}$ of figures 5.4 and C.7). This makes sense because volume rendering is a weighted sum of samples along a ray (see equation 4.1). If the robot moves closer to a particular region, then fewer dispensable samples influence the summation. Other false-positive predictions are located closer to the robot but sideways to its trajectory where few training data is collected (e.g. $x = 0\,\text{m}$, $y = 3\,\text{m}$ in figure 5.4). Predicting false-positive objects is less critical for local tasks, e.g. collision avoidance, compared to global path planning which would suffer from blocked routes.

Occupancy Grid

The occupancy grids are a rough approximation of the occupied area. As seen in figure 5.7, the free space in the middle of the *office* is separated from the nearby objects. Some areas are considered to be empty despite being occupied in reality, e.g. the wall at ($x = -1\,\text{m}$, $y = 4\,\text{m}$) in the *office*. These fields are normally lateral to the trajectory of the robot and due to the sensor setup only a few depth measurements are made there. In this case, sampling points along depth measurements to update the occupancy grid is not sufficient. This drawback could be addressed either by adding sensors pointing to the left and right of the robot or by randomly sampling points for the *NeRF-Update* like in *Instant-NGP* instead of along depth measurements (see chapter 4.3.4).

Regarding the NND, the occupancy grid of *VIRUS-NeRF* is compared to the one of *Instant-NGP* in chapter 5.5.2. On average, *VIRUS-NeRF* makes 11.64 training steps per second in contrast to only 7.96 when using the grid of *Instant-NGP*. This speed-up of 46% can be explained by two factors: *Instant-NGP* samples during the first 256 training steps all 128^3 grid cells and afterwards a quarter which is significantly more than 1024 samples used in *VIRUS-NeRF*. Additionally, the occupancy grid of *VIRUS-NeRF* relies partially on probabilistic sensor models (i.e. *Depth-Update*, see chapter 4.3) instead of interfering with the NeRF network which is computationally less expensive. Accelerating the training process with the occupancy grid of *VIRUS-NeRF* could be very interesting for any real-time application based on depth supervised *Instant-NGP*, e.g. *NeRF-SLAM* [47].

Ablation Study

The ablation study suggests that the combination of USSs, IRSs and cameras is beneficial. Moreover, the addition of depth sensors is necessary: the mapping breaks down if *VIRUS-NeRF* is trained uniquely with cameras or the original *Instant-NGP* [40] is used instead. This may be unexpected because standard NeRF as well as *Instant-NGP* is solely based on images and other variants can represent large environments [42], [43], [46]. However, the dataset of this project has sparse measurements and a low view diversity compared to most other datasets. For example, *iMap* [18], *Nice-SLAM* [19] and *NeRF-SLAM* [47] are assessed on the indoor scenes of the *Replica* dataset [67] sampling twice as many images along a random trajectory. The dataset of *VIRUS-NeRF* contains pairs of images taken simultaneously with an overlapping FoV and by a cumbersome and grounded robot having a limited range of motion. Therefore, the images of *Replica* are characterized by significantly greater variations in viewing perspectives. When removing the IRS, the results of *VIRUS-NeRF* are inferior compared to eliminating the USS, probably because the *NeRF-Update* of the occupancy grid becomes incomplete due to sampling along narrow and sparse IRS measurements instead of wide USS ones. This phenomenon can be observed by comparing figures C.14 and C.15 which show occupancy grids updated without USS and IRS respectively. It is not surprising that the RGB-D camera outperforms all other sensors because its depth images are more dense and more accurate than low-cost alternatives.

The occupancy grid of *VIRUS-NeRF* improves the NND compared to the grid of *Instant-NGP*, especially regarding median and inlier metrics. Further, the occupancy grid of *VIRUS-NeRF* is bounded by $[0, 1]$ having a probabilistic meaning and being a widely known concept in robotics. The grid of *Instant-NGP* contains values up to infinity which is harder to interpret.

Optimizing the poses with bundle adjustment is helpful, especially for large environments because one LiDAR scan observes a smaller portion of the scene. The ablation study shows that the hyper-parameter optimization does indeed improve the mapping (see table C.1). However, the parameters do not generalize well to other environments, e.g. *common area* (see table 5.1). A closer investigation is required to understand if the PSO overfits the occupancy grid parameters or the ones related to the training process. For particle 1 (second-best PSO neighbourhood), the gap to the hand-tuned parameters gets smaller showing the ability of PSO to produce different candidate solutions.

5.6 Training Speed

5.6.1 Results

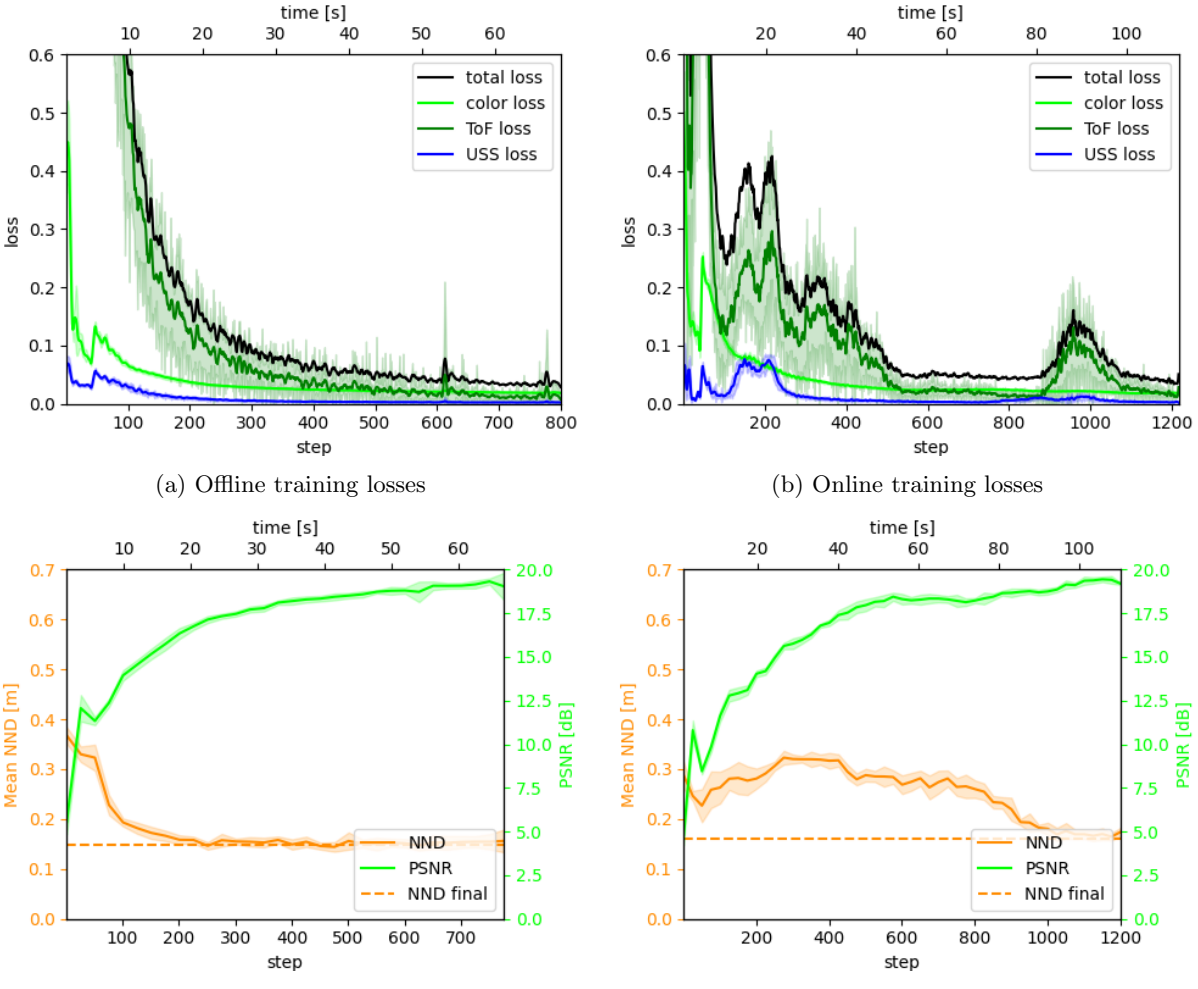
Online operation differs from offline training by having only the previously taken measurements available, in contrast to all training data of one scene. The real-time simulation lasts for the duration of the experiment and is evaluated exclusively on visited poses. All training is done on a *Nvidia Titan Xp* GPU.

Figure 5.8 shows the average losses and metrics over 10 runs for the *office*. Looking at the loss curves, the offline training is smoother than the online one. For offline operation, the NND converges already after 20 seconds whereas online after 90 seconds. Similarly, the *Peak Signal to Noise Ratio* (PSNR) converges online much slower. The final NND for online learning is 0.161 m compared to 0.148 m in the offline case. Figure C.16 presents the same plots for the *common area*. For online simulation, the model seems to converge first similarly to the *office*, then the NND and the PSNR worsen again during the last third of the training process. As a consequence, the final online NND of 0.599 m is much worse compared to 0.403 m in offline learning.

5.6.2 Discussion

The *Taichi implementation* of Instant-NGP used for *VIRUS-NeRF* is about 20% slower than the original one [55]. *VIRUS-NeRF* converges offline after approximately 20 seconds. Even with a boost of 20% this is not fast enough for real-time performance. The computational speed, the training data and the algorithm are possible limitations of the convergence speed. It seems that the available data is more important compared to the computational power because, despite making more training steps, the online algorithm converges much slower than the offline one. For online simulation, the NND starts to converge after half of the training process in the *office* environment being approximately the moment when the robot turns around and drives back towards its starting position. These observations suggest that a higher variety of viewpoints improves the convergence significantly and it is in line with most NeRF algorithms that rely on a plurality of images [37].

In the *common area*, the degradation of the metrics occurs approximately when the robot makes a sharp turn and starts moving to an unseen part of the scene (see figure C.16). Most likely, the NND would recover if the training was continued. Contrary to the *office*, the USS loss is more important than the IRS loss (*ToF loss*) in the *common area*. One possible cause is that in the *common area* the objects are more spread out than in the *office*. This leads to 35.7% of all IRS measurements being valid in the *office* and only 10.3% in the *common area*. Therefore, *VIRUS-NeRF* is trained using fewer IRS samples and the USS becomes more important. This is a disadvantage of the IRS having a maximum range of 4 m.



(a) Offline training losses

(b) Online training losses

(c) Offline training metrics

(d) Online training metrics

Figure 5.8: *Office* training curves: The coloured area indicates the standard deviation over 10 runs. The mean losses are smoothed with a Savitzky-Golay filter (order 3, window size 10). The NND is the accuracy ($Sensor \rightarrow GT$) in zone 3 ($0 - 100\,\text{m}$).

Chapter 6

Conclusion

This study presents *VIRUS-NeRF - Vision, InfraRed and UltraSonic based Neural Radiance Fields* for local mapping. Multiple low-cost sensors are analysed and three *UltraSonic Sensors* (USS) are tested extensively: The *URM37* is the most accurate and most robust sensor for various objects and measurement distances. The final sensor arrangement contains two stacks of an *InfraRed Sensor* (IRS), a USS and a camera mounted on a robotic platform. The attempt to synchronize the sensors must be revised. The IRS may need to be replaced by a device that can be triggered if the robot is to move faster. In any case, the beam width of the IRS should be determined experimentally instead of being optimised as a hyper-parameter.

To evaluate *VIRUS-NeRF*, a dataset is collected at ETHZ and *VIRUS-NeRF* is compared to instantaneous scans of USSs, IRSs and LiDARs using a global map created by LiDAR point clouds. The results show that *VIRUS-NeRF* has comparable coverage to a LiDAR and is much better than USSs and IRSs. The accuracy depends on the environment: In a smaller room with optimized hyper-parameters (*office*), the performance of *VIRUS-NeRF* is equal to a LiDAR. In larger rooms with no optimized hyper-parameters (*common area* or *corridor*), the accuracy worsens due to the limited reach of the IRS and it is close to the results of USSs. These findings demonstrate the primary objective of the project: it is possible to use low-cost sensors and achieve high performance in local indoor mapping, i.e. comparable to LiDAR point clouds, especially in applications mainly requiring good coverage.

The ablation study shows that adding the USS and the IRS to the camera is important. Only the much more expensive RGB-D camera outperforms this sensor configuration and removing either sensor degrades the results substantially. If the maps are learned without any depth sensor or with the original *Instant-NGP* [40], they deteriorate even further. The occupancy grid of *VIRUS-NeRF* used for ray marching improves all metrics over the one from *Instan-NGP* and its updating speed is about 46% faster. In addition, the occupancy grid of *VIRUS-NeRF* is bounded by $[0, 1]$ having a probabilistic meaning and being more intuitive. The optimization of the poses through bundle adjustment is necessary, especially in larger environments (*common area*). By using *Particle Swarm Optimization* (PSO), the hyper-parameters can be tuned for one particular environment.

VIRUS-NeRF has mainly three limitations: First, the accuracy is limited. Second, the hyper-parameters optimized in one scene do not generalize well to other ones. Third, the model converges not fast enough for real-time operation, particularly if the measurements are taken online. These drawbacks can be addressed by following improvements:

- **Accuracy**: The cube-shaped hash table and occupancy grid are useful for scenes that extend evenly in all directions. However, indoor environments are usually flat and the data structure could be adapted accordingly.

- **Accuracy**: The standard deviation of the IRS measurements could be used for loss weighting. *NeRF-SLAM* shows that uncertainty weighting of the loss has an important impact on the PSNR [47].
- **Accuracy**: Multiple NeRF predictions could be averaged to make the local mapping more robust by varying the height and centre of the depth scans.
- **Accuracy/Speed**: Comparing the offline and online training shows that lots of data with large view variations are crucial for fast convergence and accuracy. Preferably, the robot has sensors pointing in all directions. In particular, lateral IRSs could capture more data than longitudinal ones, as the space in front of the robot is unoccupied when moving in straight lines. (If the FoVs of the cameras no longer overlap, the extrinsic camera-to-camera calibration must be performed differently.)
- **Accuracy/Speed**: Similar to *Neuralangelo* [46], a coarse-to-fine scheme could prevent convergence to suboptimal local minima by gradually increasing the number of hash grids. To match online learning, the scheme should depend on the sampling density of a given region rather than the training time.
- **Speed**: The *NeRF-Update* of the occupancy grid could be accelerated by using the same predictions that were already used for training.
- **Speed**: *Instant-NGP* has two *Multi-Layer Perceptrons* (MLPs), the first of which predicts the density and the second the colour, with the density serving as input. Only the density is required for the depth prediction. By pre-training (or alternately training) the density MLP without the colour MLP, the convergence speed could be increased.
- **Speed/PSO**: Using the cumulative score of intermediate training steps for PSO instead of the final one would promote a faster rate of convergence during optimization.
- **PSO**: The hyper-parameters could be evaluated at each iteration in multiple environments updating the PSO based on the average score. This is expected to produce more general solutions and be less prone to "lucky" one-time evaluations.
- **PSO**: The trade-off between exploration and exploitation could be driven by the evaluation variance of the best position of the particles. If the variance is low, more computing resources are allocated for exploration and vice versa.

This research shows that *VIRUS-NeRF* is an effective method for local mapping based on a low-cost sensor setup. Although the algorithm is assessed on 2D scans, *VIRUS-NeRF* learns the density distribution across the entire 3D environment enabling global mapping. To facilitate compatibility with established methodologies the density can be mapped from $[0, \infty)$ (MLP output) to $[0, 1]$ as established in the *NeRF-Update* of *VIRUS-NeRF*. This global representation may exhibit reduced accuracy due to the optimization focus on depth prediction via volume rendering rather than directly on the density output of the MLP. Nevertheless, such a coarse scene representation could prove adequate for global path planning tasks, while local scans, as demonstrated in this work, remain essential for safety-critical functions like collision avoidance. Moreover, dynamic scenes could be addressed by decoupling static and transient components, akin to the approach seen in *NeRF in the Wild* [38]. Finally, *VIRUS-NeRF* could incorporate pose prediction solving the full SLAM problem similar to *iMap* [18], *Nice-SLAM* [19] or *NeRF-SLAM* [47].

Bibliography

[1] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *Proceedings. 1985 IEEE international conference on robotics and automation*, vol. 2. IEEE, 1985, pp. 116–121.

[2] L. Matthies and A. Elfes, "Integration of sonar and stereo range data using a grid-based representation," in *Proceedings. 1988 IEEE International Conference on Robotics and Automation.* IEEE, 1988, pp. 727–733.

[3] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[4] S. Tijmons, G. C. H. E. de Croon, B. D. W. Remes, C. De Wagter, and M. Mulder, "Obstacle avoidance strategy using onboard stereo vision on a flapping wing mav," *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 858–874, 2017.

[5] K. McGuire, G. de Croon, C. De Wagter, K. Tuyls, and H. Kappen, "Efficient optical flow and stereo vision for velocity estimation and obstacle avoidance on an autonomous pocket drone," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1070–1076, 2017.

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

[7] A. Masoumian, H. A. Rashwan, J. Cristiano, M. S. Asif, and D. Puig, "Monocular depth estimation using deep learning: A review," *Sensors*, vol. 22, no. 14, p. 5353, 2022.

[8] X. Dong, M. A. Garratt, S. G. Anavatti, and H. A. Abbass, "Towards real-time monocular depth estimation for robotics: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 16 940–16 961, 2022.

[9] T. Ehret, "Monocular depth estimation: a review of the 2022 state of the art," *Image Processing On Line*, vol. 13, pp. 38–56, 2023.

[10] 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.

[11] 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.

[12] Z. Xie, X. Yu, X. Gao, K. Li, and S. Shen, "Recent advances in conventional and deep learning-based depth completion: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[13] M. Schlatter, "Image guided depth completion using ultrasonic range measurements," 2022.

[14] M. Rubio, "Ultrasonic sensor fusion for local safety of mobile robots," 2023.

[15] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[16] K. Rematas, A. Liu, P. P. Srinivasan, J. T. Barron, A. Tagliasacchi, T. Funkhouser, and V. Ferrari, "Urban radiance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 932–12 942.

[17] A. Carlson, M. S. Ramanagopal, N. Tseng, M. Johnson-Roberson, R. Vasudevan, and K. A. Skinner, "Cloner: Camera-lidar fusion for occupancy grid-aided neural representations," *IEEE Robotics and Automation Letters*, vol. 8, no. 5, pp. 2812–2819, 2023.

[18] E. Sucar, S. Liu, J. Ortiz, and A. J. Davison, "imap: Implicit mapping and positioning in real-time," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6229–6238.

[19] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, "Nice-slam: Neural implicit scalable encoding for slam," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 786–12 796.

[20] A. Elfes, "A tesselated probabilistic representation for spatial robot perception and navigation," in *JPL, California Inst. of Tech., Proceedings of the NASA Conference on Space Telerobotics, Volume 2*, 1989.

[21] K. Konolige, "Improved occupancy grids for map building," *Autonomous Robots*, vol. 4, pp. 351–367, 1997.

[22] S. Thrun, "Learning occupancy grid maps with forward sensor models," *Autonomous robots*, vol. 15, pp. 111–127, 2003.

[23] T. Collins and J. Collins, "Occupancy grid mapping: An empirical evaluation," in *2007 mediterranean conference on control & automation*. IEEE, 2007, pp. 1–6.

[24] C. Coué, C. Pradalier, C. Laugier, T. Fraichard, and P. Bessière, "Bayesian occupancy filtering for multitarget tracking: an automotive application," *The International Journal of Robotics Research*, vol. 25, no. 1, pp. 19–30, 2006.

[25] R. Danescu, F. Oniga, and S. Nedevschi, "Modeling and tracking the driving environment with a particle-based occupancy grid," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1331–1342, 2011.

[26] J. Saarinen, H. Andreasson, and A. J. Lilienthal, "Independent markov chain occupancy grid maps for representation of dynamic environment," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 3489–3495.

[27] D. Meyer-Delius, M. Beinhofer, and W. Burgard, "Occupancy grid models for robot mapping in changing environments," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 26, no. 1, 2012, pp. 2024–2030.

[28] R. Garg, V. K. Bg, G. Carneiro, and I. Reid, "Unsupervised cnn for single view depth estimation: Geometry to the rescue," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14*. Springer, 2016, pp. 740–756.

[29] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3354–3361.

[30] S. F. Bhat, I. Alhashim, and P. Wonka, "Adabins: Depth estimation using adaptive bins," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4009–4018.

[31] L. Wang, M. Famouri, and A. Wong, "Depthnet nano: A highly compact self-normalizing neural network for monocular depth estimation," *arXiv preprint arXiv:2004.08008*, 2020.

[32] Y. Lin, T. Cheng, Q. Zhong, W. Zhou, and H. Yang, "Dynamic spatial propagation network for depth completion," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 2, 2022, pp. 1638–1646.

[33] J.-T. Lin, D. Dai, and L. Van Gool, "Depth estimation from monocular images and sparse radar data," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 233–10 240.

[34] Y. Long, D. Morris, X. Liu, M. Castro, P. Chakravarty, and P. Narayanan, "Radar-camera pixel depth association for depth completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12 507–12 516.

[35] O. Abdulaaty, G. Schroeder, A. Hussein, F. Albers, and T. Bertram, "Real-time depth completion using radar and camera," in *2022 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*. IEEE, 2022, pp. 1–6.

[36] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[37] K. Gao, Y. Gao, H. He, D. Lu, L. Xu, and J. Li, "Nerf: Neural radiance field in 3d vision, a comprehensive review," *arXiv preprint arXiv:2210.00379*, 2022.

[38] R. Martin-Brualla, N. Radwan, M. S. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, "Nerf in the wild: Neural radiance fields for unconstrained photo collections," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7210–7219.

[39] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, "Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction," *arXiv preprint arXiv:2106.10689*, 2021.

[40] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Transactions on Graphics (ToG)*, vol. 41, no. 4, pp. 1–15, 2022.

[41] M. Teschner, B. Heidelberger, M. Müller, D. Pomerantes, and M. H. Gross, "Optimized spatial hashing for collision detection of deformable objects." in *Vmv*, vol. 3, 2003, pp. 47–54.

[42] H. Turki, D. Ramanan, and M. Satyanarayanan, "Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 922–12 931.

[43] M. Tancik, V. Casser, X. Yan, S. Pradhan, B. Mildenhall, P. P. Srinivasan, J. T. Barron, and H. Kretzschmar, "Block-nerf: Scalable large scene neural view synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8248–8258.

[44] M. Oechsle, S. Peng, and A. Geiger, "Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5589–5599.

[45] J. Wang, P. Wang, X. Long, C. Theobalt, T. Komura, L. Liu, and W. Wang, "Neuris: Neural reconstruction of indoor scenes using normal priors," in *European Conference on Computer Vision*. Springer, 2022, pp. 139–155.

[46] Z. Li, T. Müller, A. Evans, R. H. Taylor, M. Unberath, M.-Y. Liu, and C.-H. Lin, "Neuralangelo: High-fidelity neural surface reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 8456–8465.

[47] A. Rosinol, J. J. Leonard, and L. Carlone, "Nerf-slam: Real-time dense monocular slam with neural radiance fields," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 3437–3444.

[48] C.-M. Chung, Y.-C. Tseng, Y.-C. Hsu, X.-Q. Shi, Y.-H. Hua, J.-F. Yeh, W.-C. Chen, Y.-T. Chen, and W. H. Hsu, "Orbeez-slam: A real-time monocular visual slam with orb features and nerf-realized mapping," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9400–9406.

[49] Z. Teed and J. Deng, "Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras," *Advances in neural information processing systems*, vol. 34, pp. 16 558–16 569, 2021.

[50] C.-H. Lin, J. Gao, L. Tang, T. Takikawa, X. Zeng, X. Huang, K. Kreis, S. Fidler, M.-Y. Liu, and T.-Y. Lin, "Magic3d: High-resolution text-to-3d content creation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 300–309.

[51] L. Melas-Kyriazi, I. Laina, C. Rupprecht, and A. Vedaldi, "Realfusion: 360deg reconstruction of any object from a single image," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 8446–8455.

[52] V. Niculescu, T. Polonelli, M. Magno, and L. Benini, "Nanoslam: Enabling fully onboard slam for tiny robots," *IEEE Internet of Things Journal*, 2023.

[53] STMicroelectronics, "Vl53l5cx infrared time-of-flight ranging sensor," https://www.st.com/en/imaging-and-photonics-solutions/vl53l5cx.html.

[54] Intel, "Realsense depth camera d435," https://www.intelrealsense.com/depth-camera-d435/.

[55] T. Lang, "Taichi lang: Instant-ngp implementation," https://docs.taichi-lang.org/blog/taichi-instant-ngp, accessed on 2024-01-24.

[56] J. R. Ruiz-Sarmiento, C. Galindo, and J. González-Jiménez, "Robot@ home, a robotic dataset for semantic mapping of home environments," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 131–141, 2017.

[57] I. Bots, "Inspector bots," https://www.inspectorbots.com/, accessed on 2024-01-16.

[58] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1280–1286.

[59] P. Furgale, T. D. Barfoot, and G. Sibley, "Continuous-time batch estimation using temporal basis functions," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 2088–2095.

[60] J. Maye, P. Furgale, and R. Siegwart, "Self-supervised calibration for robotic systems," in *2013 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2013, pp. 473–480.

[61] L. Oth, P. Furgale, L. Kneip, and R. Siegwart, "Rolling shutter camera calibration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1360–1367.

[62] D. Tsai, S. Worrall, M. Shan, A. Lohr, and E. Nebot, "Optimising the selection of samples for robust lidar camera calibration," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 2631–2638.

[63] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss, "KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way," *IEEE Robotics and Automation Letters (RA-L)*, vol. 8, no. 2, pp. 1029–1036, 2023.

[64] Z. Liu, X. Liu, and F. Zhang, "Efficient and consistent bundle adjustment on lidar point clouds," *IEEE Transactions on Robotics*, 2023.

[65] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4. IEEE, 1995, pp. 1942–1948.

[66] S. Mahmud, G. Khan, M. Rahman, H. Zafar *et al.*, "A survey of intelligent car parking system," *Journal of applied research and technology*, vol. 11, no. 5, pp. 714–726, 2013.

[67] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma *et al.*, "The replica dataset: A digital replica of indoor spaces," *arXiv preprint arXiv:1906.05797*, 2019.

[68] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, pp. 189–206, 2013.

[69] D. Meagher, "Geometric modeling using octree encoding," *Computer graphics and image processing*, vol. 19, no. 2, pp. 129–147, 1982.

[70] H. Oleynikova, A. Millane, Z. Taylor, E. Galceran, J. Nieto, and R. Siegwart, "Signed distance fields: A natural representation for both mapping and planning," in *RSS 2016 workshop: geometry and beyond-representations, physics, and scene understanding for robotics*. University of Michigan, 2016.

[71] 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*, 2011, pp. 127–136.

[72] H. Pfister, M. Zwicker, J. Van Baar, and M. Gross, "Surfels: Surface elements as rendering primitives," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000, pp. 335–342.

[73] J. Kläß, J. Stückler, and S. Behnke, "Efficient mobile robot navigation using 3d surfel grid maps," in *ROBOTIK 2012; 7th German Conference on Robotics*. VDE, 2012, pp. 1–4.

[74] J. Stückler and S. Behnke, "Multi-resolution surfel maps for efficient dense 3d modeling and tracking," *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, pp. 137–147, 2014.

[75] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "Elasticfusion: Real-time dense slam and light source estimation," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1697–1716, 2016.

[76] M. Schadler, J. Stückler, and S. Behnke, "Multi-resolution surfel mapping and real-time pose tracking using a continuously rotating 2d laser scanner," in *2013 ieee international symposium on safety, security, and rescue robotics (ssrr)*. IEEE, 2013, pp. 1–6.

[77] C. Park, S. Kim, P. Moghadam, C. Fookes, and S. Sridharan, "Probabilistic surfel fusion for dense lidar mapping," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 2418–2426.

[78] M. Poggi, F. Aleotti, F. Tosi, and S. Mattoccia, "On the uncertainty of self-supervised monocular depth estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3227–3237.

[79] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in neural information processing systems*, vol. 30, 2017.

[80] J. Hornauer and V. Belagiannis, "Gradient-based uncertainty for monocular depth estimation," in *European Conference on Computer Vision*. Springer, 2022, pp. 613–630.

[81] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Berlin, Heidelberg: Springer-Verlag, 2007.

[82] Z. Qiu, Y. Lu, and Z. Qiu, "Review of ultrasonic ranging methods and their current challenges," *Micromachines*, vol. 13, no. 4, 2022.

[83] R. Kapoor, S. Ramasamy, A. Gardi, R. van Schyndel, and R. Sabatini, "Acoustic sensors for air and surface navigation applications," *Sensors*, vol. 18, p. 499, 02 2018.

[84] T. InvenSense, "Ch201 sensor evaluation model," https://product.tdk.com/en/search/sensor/ultrasonic/tof/info?part_no=EV_MOD_CH201-00-01, accessed on 2023-08-10.

[85] SlamTec, "Rplidar a1," https://www.slamtec.com/en/Lidar/A1/.

[86] S. S. Inovation, "Microscan 3," https://www.sick.com/fr/en/safety-laser-scanners/safety-laser-scanners/microscan3/mics3-cbaz55za1/p/p586547.

[87] Y. Ming, X. Meng, C. Fan, and H. Yu, "Deep learning for monocular depth estimation: A review," *Neurocomputing*, vol. 438, pp. 14–33, 2021.

[88] MaxBotix, "Best ultrasonic sensor for indoor mobile robotics," https://maxbotix.com/blogs/blog/best-ultrasonic-sensor-indoor-mobile-robotics.

[89] Y. Jin, S. Li, J. Li, H. Sun, and Y. Wu, "Design of an intelligent active obstacle avoidance car based on rotating ultrasonic sensors," in *2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, 2018, pp. 753–757.

Appendix A

Literature Review

A.1 Map Representations

In sensor fusion, different measurements must be converted to a common representation. A common approach is to create a map of the environment that is updated in an online fashion and helps to filter noisy data. In general, sparse and dense maps can be distinguished: Sparse maps register only some particular features of the environment, e.g. landmarks, beacons or the position of particular objects. Sparse maps are useful for localization. However, in most environments, no landmarks or beacons are present and object-based maps have the problem of data association [21]. Dense maps divide the environment into many small cells which contain some local information. Dense maps are more suited for tasks like collision avoidance.

Occupancy grids are conceptually simple and well-established in academia and industry (see chapter 2.2). However, in three-dimensional space, occupancy grids get very memory-expensive. Therefore, more sophisticated data structures are developed like the *OctoMap* [68]. *OctoMaps* are based on the hierarchical tree-like structure of octrees [69] where every parent node contains 8 child nodes. Each child node represents a voxel (volume pixel in 3D space) and carries information about its occupancy. The minimum resolution of the map is defined by the depth of the graph. If all leaf nodes of one parent node are equal and identified with high confidence, they are pruned to save memory. This allows an *OctoMap* to be much more compressed than a 3D occupancy grid.

An alternative to *OctoMaps* is the *Signed Distance Field* (SDF). An SDF contains two pieces of information about each voxel: the distance to the next surface as well as its uncertainty [70]. The map is *signed* because the voxel is positive if it is outside an object, negative if it is inside an object and equal to zero if it is on the boundary. Even though SDFs are discretized, they model continuous functions. Therefore, the resolution of the map is higher than the pure number of voxels. This allows for reducing the memory requirements. The *distance* can be defined in different ways: *Euclidean Signed Distance Fields* (ESDF) use the closest Euclidean distance from a voxel to the surface and are useful in obstacle circumnavigation. *Truncated Signed Distance Fields* (TSDF) measure the distance between the voxel and the surface along a one-dimensional sensor ray. TSDFs allow a fast reconstruction from depth camera data. They gain much popularity through their use in *KinectFusion* [71].

Other 3D representation methods are based on *Surfels* (surface elements) that have been used for a long time in 3D rendering [72]. *Surfels* hold certain attributes that approximate locally the surface of an object. These attributes are estimated for a certain voxel by continuous measurements and a probabilistic model. Attributes can represent the 3D position, occupancy, surface normal or colour. For example, the surface normal can be estimated by the eigenvector of the smallest eigenvalue from the position co-

variance [73]. This approach avoids discretization effects by using continuous statistics inside each voxel. Similar to *OctoMaps*, *Surfel* maps can be structured with octrees where each parent node contains the statistical properties of its child nodes [74]. This allows a fast sampling at any resolution. In robotics, *Surfel* maps are often used with depth cameras, e.g. *ElasticFusion* [75]. However, the method is also applied using 2D [76] and 3D [77] LiDAR sensors.

A.2 Uncertainty in Monocular Depth Estimation

In robotics, most fusion and mapping techniques are probabilistic. Normally, MDE models do not contain any information about the uncertainty of the prediction. The uncertainty may be estimated through empirical and predictive estimation [78]: Empirical methods aim to explain the model uncertainty e.g. epistemic uncertainty. Thereby, they measure the variance between a set of all the possible network configurations. For example, the *Monte Carlo Dropout* technique (normally used to prevent over-fitting during training) can be applied during testing to calculate the distribution of the output from slightly different models. *Bootstrapped Ensemble* is a different method that trains multiple and randomly initialised networks on a subset of the entire training set [79]. Then, the uncertainty can be estimated by analysing the variance of the model's output. Predictive methods try to learn the aleatoric uncertainty i.e. inherent randomness of the data. For example, using a log-likelihood maximization, the data mean and variance can be deduced from its distribution [78].

The drawback of all these methods is that either the model must be modified which prohibits taking existing networks, or several forward passes are required to get meaningful statistics. One approach that circumvents these problems makes two forward passes with the original image and its right-left flipped version [80]. Then, the output of the transformed image is flipped back and the error between the two outputs is calculated. This error is back-propagated for some layers and converted to the pixel-vice uncertainty. This approach seems to outperform the before-mentioned methods while no additional training and only two forward and one backward passes are required [80].

Appendix B

Sensors

B.1 Ultrasonic Sensor

B.1.1 General

UltraSonic Sensors (USSs) use ultrasound waves to measure the distance to an object. Ultrasound refers to sound waves above human hearing i.e. above 20kHz. The following list presents the main advantages and limitations of USS [81]. The attributes are discussed in more detail in the preceding sections:

Advantages:

- Low computational effort
- Large coverage
- Low power consumption
- Material recognition
- Lightweight
- Low cost

Disadvantages:

- Low sampling frequency
- Low angular resolution
- Limited radial resolution
- Material dependent
- Measurement artefacts
- Cross-Talking

B.1.2 Time of Flight

USSs have an emitter which sends out the sound wave and a receiver which registers its echo. In some cases, the transceiver and the receiver are designed in one piece [81]. Most USSs in the lower price segment ($< 50CHF$) use the *Time of Flight* (ToF) method to measure the distance:

$$d = \frac{ct}{2} \tag{B.1}$$

where d is the distance from the sensor to the closest object, t is the elapsed time between sending out and receiving the signal and c is the speed of sound in the air. (For more advanced techniques like the two-frequency continuous wave method, the binary frequency shift keying method or the amplitude modulation method see [82].)

Most USSs consider only the first echo signal registering the closest object in the FoV. Often, the echo is detected if it exceeds a certain threshold [82], The amplitude threshold is about 3 to 5 times larger than the noise level to reject disturbances. This method always exhibits an error because the echo must first exceed the amplitude before triggering a measurement. This inaccuracy depends on the strength of the echo which depends

on the distance to the obstacle. To prevent this error the correlation between the transmitted and the echo signal can be calculated. Using the maximum cross-correlation the time is evaluated more precisely than by using the threshold method. However, this method requires more computation and therefore, it is slower.

B.1.3 Sampling Frequency

Considering a USS with a maximum range of 5 m, the sound travels up to 29 ms assuming a speed of sound of $c = 343.3\,\mathrm{m\,s^{-1}}$. This limits the sample rate to approximately 34 Hz. Normally, the USS emits several pulses to achieve greater robustness and multiple reflections may occur in complex environments which adds some delay. In practice, most sensors have a sample rate between 10 and 25 Hz (see market search, table 3.1). The following chapters discuss the influence of the temperature and the wave frequency on the sampling frequency.

B.1.4 Speed of Sound

The distance to an object is proportional to the ToF of the sound wave assuming that the speed of sound is known (see equation B.1). However, the speed of sound varies with the air temperature, pressure and humidity [81]. The following equations describe the dependency:

$$c = 20.05\sqrt{\frac{T_C + 273.16}{1 - 3.79 * 10^{-3} * h_r * \frac{p_{sat}}{p_s}}}[\frac{m}{s}] \tag{B.2}$$

where T_c is the temperature in degree Celsius, h_r is the relative humidity, p_s is the atmospheric pressure and p_{sat} is the saturation pressure.

Consider the distance to an object in 1 meter is measured under different extreme conditions: Assuming a temperature of 20 °C and standard pressure (101325 Pa), the real speed of sound at low humidity ($h_r = 0.1$) is $c_{low} = 343.2965\,\mathrm{m\,s^{-1}}$ and at high one ($h_r = 1.0$) is $c_{high} = 343.3148\,\mathrm{m\,s^{-1}}$. Therefore, if the measurement is done in an environment with very high humidity c_{high} but c_{low} is used instead, the resulting error is about 53 µm. Similarly, fixing the temperature at 20 °C and the relative humidity at $h_r = 0.3$, at sea level the speed of sound is $c_{sea} = 343.3005\,\mathrm{m\,s^{-1}}$ and on Mount Everest $c_{everest} = 343.3142\,\mathrm{m\,s^{-1}}$. Hence, if the same experiment is accidentally conducted on Mount Everest while assuming sea level pressure, this leads to an error of about 40 µm. The accuracy of all low-cost USSs is in the millimetre range. Therefore, the influence of the air humidity and the air pressure can be ignored and the equation B.2 is approximated as follows:

$$c = 20.05\sqrt{T_C + 273.16}[\frac{m}{s}] \tag{B.3}$$

However, it makes sense to compensate temperature variations for accurate applications because $c_{sea} = 343.3\,\mathrm{m\,s^{-1}}$ at 20 °C and $c_{sea} = 331.4\,\mathrm{m\,s^{-1}}$ at 0 °C, leading to an error of about 3.6 cm for an obstacle 1 m away from the sensor. For the rest of the report, a temperature of 20 °C is considered

B.1.5 Wave Frequency

In general, the attenuation of a planar sound wave at a distance x and an initial pressure P_0 is given by the following equation [83]:

$$P = P_0 e^{-\frac{\alpha x}{2}} \tag{B.4}$$

α is the attenuation coefficient and it depends on the frequency, the temperature, the humidity and the pressure. For ultrasonic sound waves, α is approximately proportional to the squared frequency [83]. Assuming an air temperature of 20 °C, a relative humidity of 20% and atmospheric pressure, $\alpha \cong 0.09$ for 40kHz and $\alpha \cong 0.6$ for 175kHz. Considering an obstacle in 1 m distance and ignoring all losses regarding the reflection of the sound wave, $\frac{P}{P_0} = 91.4\%$ at 40kHz and $\frac{P}{P_0} = 54.9\%$ at 175kHz. This shows that USSs with lower wave frequency have a larger range but lower sampling frequency because secondary reflections are more common. Most low-cost USSs have a frequency of about 40kHz (see market search, table 3.1).

B.1.6 Echo Pattern

The sound wave propagation depends on the aperture, the wavelength, the wave frequency and other factors [81]. Figure B.1 shows the sound beam pattern of the *CH201* sensor from *TDK InvenSense* (assembled with a 45° FoV housing) [84]. The echo is measured by exposing the sensor for different angles to a $1\,m^2$ target at a distance of 30 cm. The results are typical for USSs: The echo amplitude has multiple maxima which can be described by a Bessel function in the far field (further away than $\frac{aperture^2}{\lambda}$). Normally, the FoV is approximated to have a cone shape and refers to the region where the amplitude is larger than −6 dB, e.g. approximately 45° for the CH201 sensor. A large FoV means that the sensor has a low angular resolution but a high coverage.

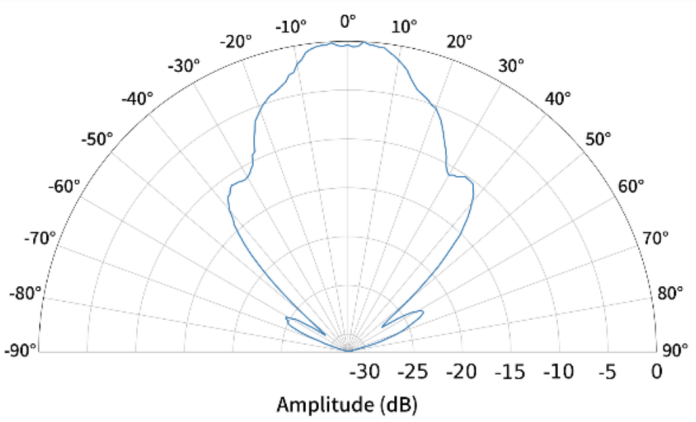


Figure B.1: Sound wave pattern of the *CH201* sensor from *TDK InvenSense* assembled with a 45° FoV housing [84]. The echo amplitude is given in dB and it is normalized to the highest amplitude.

B.2 LiDAR

Light Detection And Ranging (LiDAR) sensors emit a laser beam and are based on ToF measurements. The sensor is put on a rotating shaft to scan the surroundings in 2D or 3D. LiDAR sensors can sample faster than USS because the speed of light is much faster than the speed of sound.

One of the cheapest products on the market is the *RPlidar A1* from *SlamTec* [85]. It costs around 100 CHF on *AliExpress*, has a range of 12 m and samples at over 8kHz resulting in a scanning frequency of 5.5Hz. Its measurement accuracy of 1% is comparable to cheap USS (see table 3.1) but the angular resolution of less than 1° is much better than any USS. However, the *RPlidar A1* is a 2D LiDAR which is in most environments not sufficient. 3D LiDARs are much more expensive. For example, the *microScan3* from *SICK Sensor Intelligence* costs more than 3000 CHF. It is suited for safety-relevant tasks as it fulfils different safety norms (IEC 61508, EN ISO 13849) [86].

B.3 Stereo Vision

Monocular Depth Estimation (MDE) estimates the depth of an image using deep learning (see chapter 2.3.1). Ming et al. discuss several problems of MDE in their overview [87]: Most models are large and do not meet real-time requirements. Often, the resolution is kept low to increase the accuracy. More data is required such that the models generalize better. Depth completion addresses some of these issues but requires a LiDAR sensor (see chapter 2.3.2). Two alternative methods are presented for measuring the depth by using cameras:

In stereo vision, the triangulation of features is used to determine the corresponding distance [81]. Stereo cameras are passive sensors. Therefore, they do not struggle with cross-talk and their energy consumption is low. Especially in textureless environments, it is difficult to find suitable features for triangulation and the algorithm is computationally expensive. However, stereo vision is proven to work for obstacle avoidance of a 20 g flapping-wing-drone [4] and a 40 g quad-copter [5] which have both limited payload, computational and energy resources.

A different approach is to use an infrared camera together with an infrared projector which emits a pattern. Through the distortion of this pattern, the depth of the image can be evaluated. An example of such a camera is the *RealSense D435* from *Intel* [54]. This camera has a range of 30-300 cm, an accuracy of 2% at 2 m distance and a sampling rate of 90Hz. The *D435* costs more than 300 CHF. The *D455* combines stereo vision and IR pattern projection. It addresses the limited range of the *D435* and the lack of feature detection in textureless environments of stereo vision.

B.4 Cross-Talking

B.4.1 Qualitative Experiments

Cross-talk occurs when a receiver is activated by an emitter from another sensor. This type of interference is only present when multiple active sensors operate in the same environment.

VIRUS-NeRF utilizes cameras, USSs, and IRSs. Cameras, being passive sensors, do not pose a problem. The cross-talk between multiple USSs and IRSs is assessed through two qualitative experiments: 1) Two sensor pairs measure the distance to an object pointing in the same direction. 2) One sensor pair measures the distance to an object while the other is attached to the object pointing towards the first one. The object is a flat surface approximately 30 cm away, and the sensors operate asynchronously. In the first experiment, no interference between the sensors is observed. However, this

outcome is contingent upon the object and may resemble the second experiment if its shape or texture is altered. In the second experiment, cross-talk is evident: The USS consistently underestimates the distance by half, while the IRS mostly measures the distance accurately but with some intermediate blackouts. The USS generates inaccurate measurements, but the IRS detects the errors and produces fewer samples. Consequently, IRSs can be utilized in multi-sensor or multi-robot setups, whereas USSs pose more challenges, with potential solutions discussed in the next section.

B.4.2 Multi-USS Systems

Here are some solutions to mitigate cross-talk in multi-USS systems:

- **Sensor placement**: USS orientation can be varied to prevent a FoV overlap. However, this method limits the number of sensors used and is susceptible to errors due to unpredictable sound reflections.
- **Measurement delay**: USSs can be triggered successively with a delay between measurements. However, this approach reduces the sampling frequency inversely proportional to the number of sensors.
- **Filtering**: Instead of triggering a measurement based on exceeding a certain amplitude, a specific pattern can be recognized (see chapter B.1.2). *MaxBotix* implements cross-talk suppression for sensors such as *MB1033* and *MB1603* [88]. They tested the *MB1033* by operating 10 sensors simultaneously 2 m from a target, reporting readings within a few millimetres over 97% of the time and always within 10% of the real value.
- **Rotating USS**: Similar to LiDAR, the USS can be rotated to scan the environment. Jin et al. achieved improved obstacle avoidance with a rotating USS using a small autonomous car compared to a simple multi-sensor system [89]. The *URM37* provides direct output to control a servo motor. However, integrating a rotating USS is more complex.

B.5 Ranging Sensor Comparison

Table B.1 presents a qualitative comparison of the different ranging sensors and summarizes this chapter.

	USS	IRS	LiDAR 2D	LiDAR 3D	Monocular camera	Stereo Vision (+IR)
Price ↓	Low	Low	Medium	High	Low	Medium
Range ↑	Medium	Low	Medium	High	High	High
Accuracy ↑	Medium	High	High	High	Low	High
Angle Resolution ↑	Low	Medium	High	High	High	High
Sampling Rate ↑	Low	Medium	Medium	Medium	High	High
Noise Robustness ↑	Medium	Medium	Medium	High	Low	High
Computation Complexity ↓	Low	Low	Medium	Medium	High	High

Table B.1: Ranging sensors: qualitative comparison

Appendix C

Mapping Experiments

C.1 Dataset

C.1.1 Synchronization

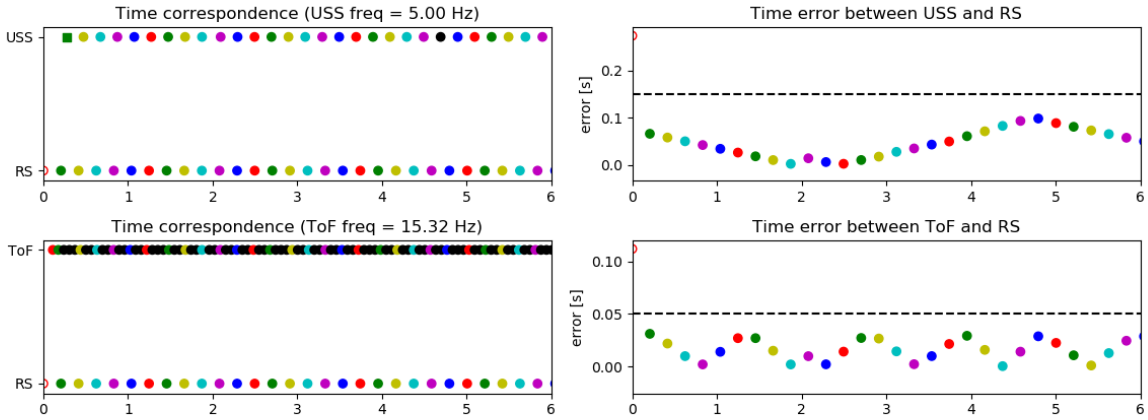


Figure C.1: During synchronization, the nearest sample in time from USS and IRS (ToF) is assigned to each measurement of the camera (RS) sampling at 4.81Hz. The x-axis is the time in seconds and shows a small scope of the sensor stack 1 from the *office* environment. The left column indicates the sample assignment using colours. Black samples are omitted and squared ones are assigned to multiple camera measurements. The right column shows the time error of the assignment where samples above a threshold are hollow and excluded from the dataset. In total, 535/538 measurements are kept in the *office*. The *common area* and *corridor* show similar results.

C.1.2 Sensor Calibration

Camera to Camera

Calibration results
==========
Camera-system parameters:
cam0 (/sync/CAM1/image_raw):
type: <class 'aslam_cv.libaslam_cv_python.DistortedPinholeCameraGeometry'>
distortion: [-0.04465574 0.03766376 0.00066069 0.0019527] +- [0.00418728 0.00414145 0.00090155 0.00093304]
projection: [388.04721481 388.36511356 327.57595477 244.29667859] +- [1.61372078 1.5528698 1.49947743 1.42859823]
reprojection error: [-0.000001, 0.000000] +- [0.106720, 0.107488]

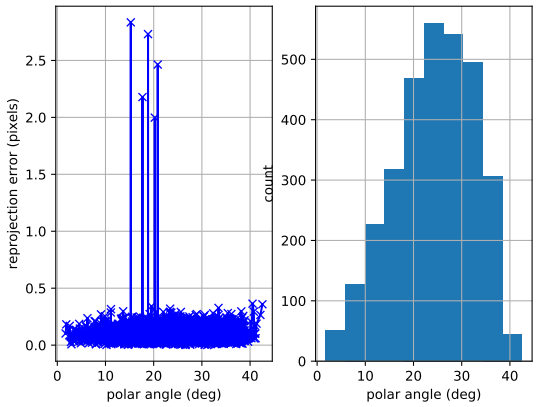
cam1 (/sync/CAM3/image_raw):
type: <class 'aslam_cv.libaslam_cv_python.DistortedPinholeCameraGeometry'>
distortion: [-0.04472216 0.03785537 0.00019454 0.00085737] +- [0.00401514 0.00447287 0.00082332 0.00087995]
projection: [387.50463035 387.69230039 334.48700779 241.91828587] +- [1.5462633 1.50105063 1.51939519 1.38525009]
reprojection error: [0.000001, -0.000000] +- [0.083003, 0.077610]

baseline T_1_0:
q: [0.00643145 0.24995226 0.01239522 0.96815746] +- [0.00376209 0.00416643 0.00110733]
t: [-0.27358526 0.00462842 -0.06490984] +- [0.0005508 0.00044053 0.00196654]

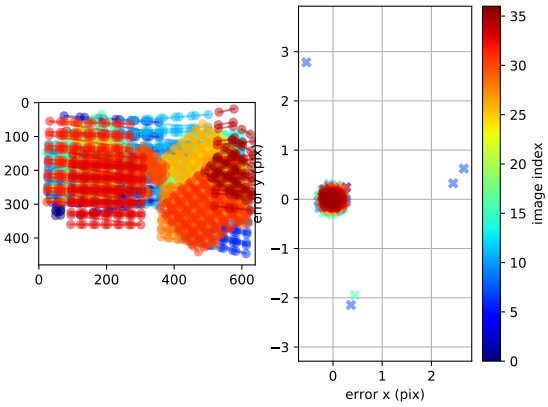
Target configuration
==========

Type: aprilgrid
Tags:
Rows: 6
Cols: 6
Size: 0.055 [m]
Spacing 0.0165 [m]

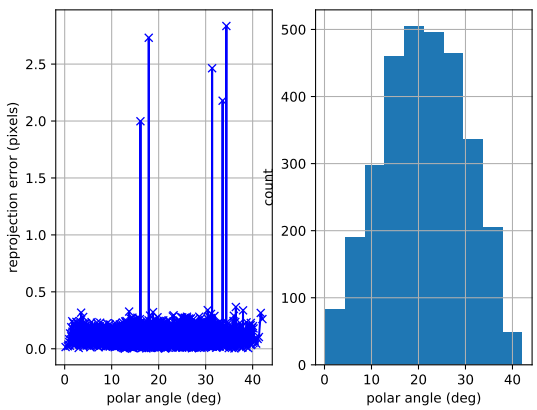
cam0: polar error



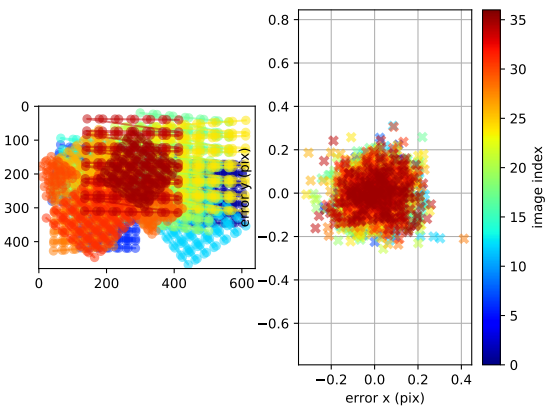
cam0: reprojection errors



cam1: polar error



cam1: reprojection errors



Camera to LiDAR

Figure C.3: Calibration board on microphone stand: The chessboard seen in this image is used for the camera-to-LiDAR calibration. The camera-to-camera calibration is based on the *Aprilgrid*.

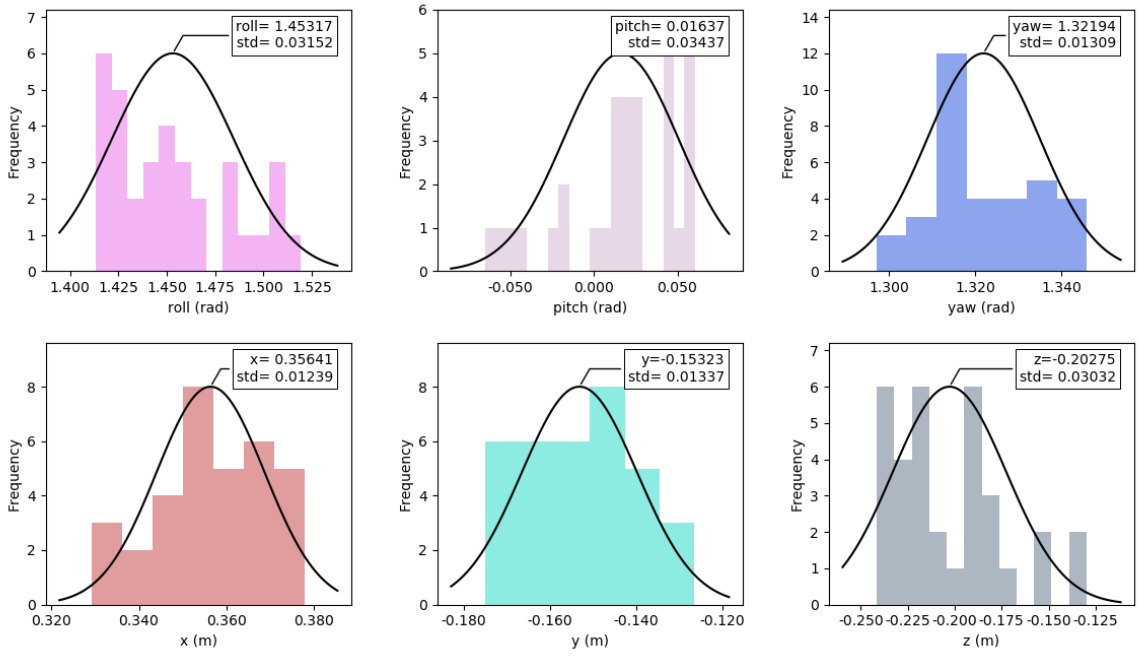


Figure C.4: Results of extrinsic *Camera-LiDAR Calibration - V 2.0* [62].

IRS to Camera

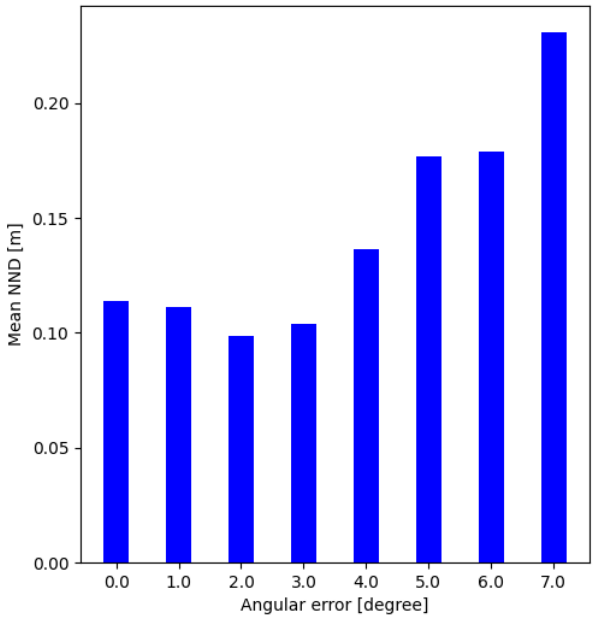


Figure C.5: Mean NND (*Sensor* $\rightarrow$ *GT*, zone 3) as a function of artificial orientation error added to the IRS. The IRS is simulated with 8x8 pixels from a RGB-D camera of the *Robot@Home2* dataset [56].

C.2 Optimization

C.2.1 Particle Swarm Optimization

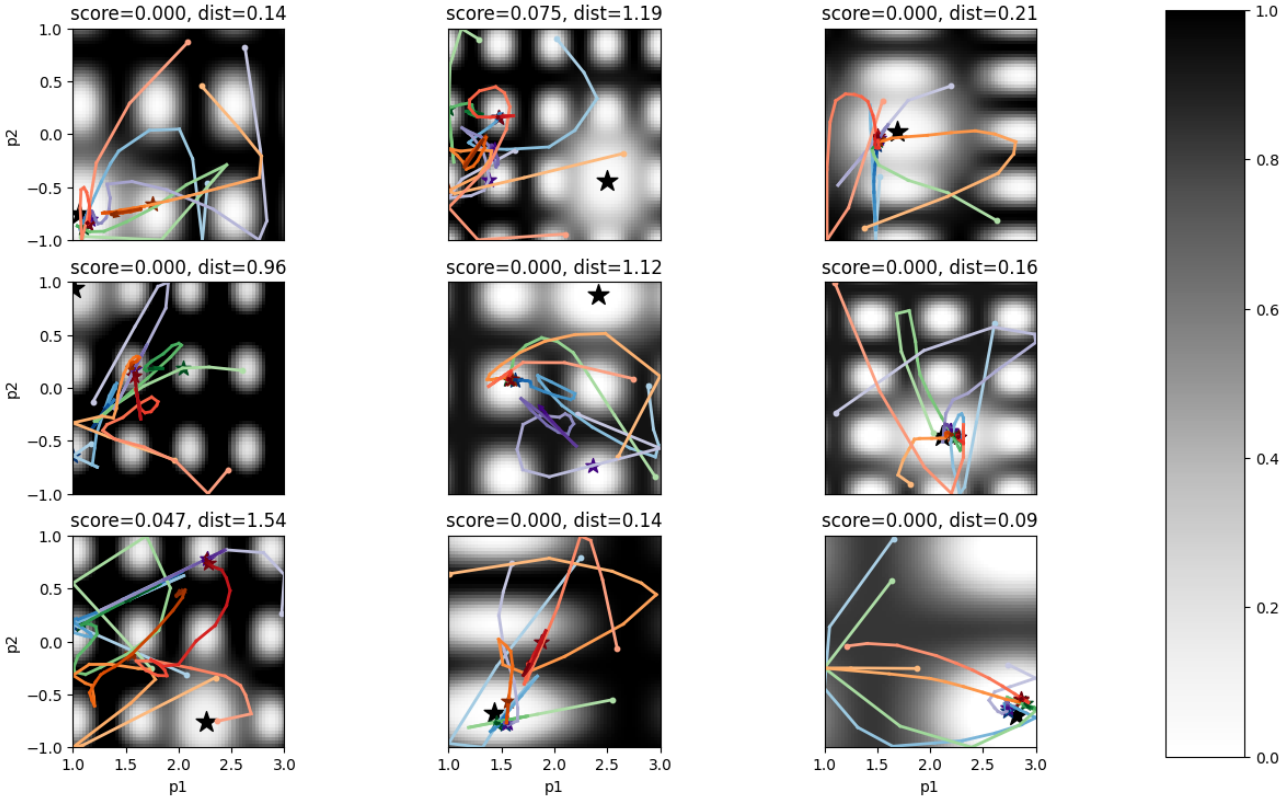


Figure C.6: 9 PSO simulations are run with 2 hyper-parameters ($p1$ and $p2$) and 5 particles. Each particle is initialized at a random position (coloured circle) and reaches the personal best position at the coloured star. The score is evaluated on the following artificial metric:
$score = r_i \frac{1+cos(2\pi \mathbf{f}*(\mathbf{x}-\mathbf{x}^0))}{2} e^{-\frac{(\mathbf{x}-\mathbf{x}^0)^2}{\sigma^2}}$ where f and σ are randomly selected in the beginning of the optimization and r_i is a random noise added to each evaluation. $\mathbf{x}^0$ is the actual best position marked with a black star. On top of the plots, there is the score of the best particle and its distance to $\mathbf{x}^0$.

C.3 Local Mapping

C.3.1 Nearest Neighbour Distance

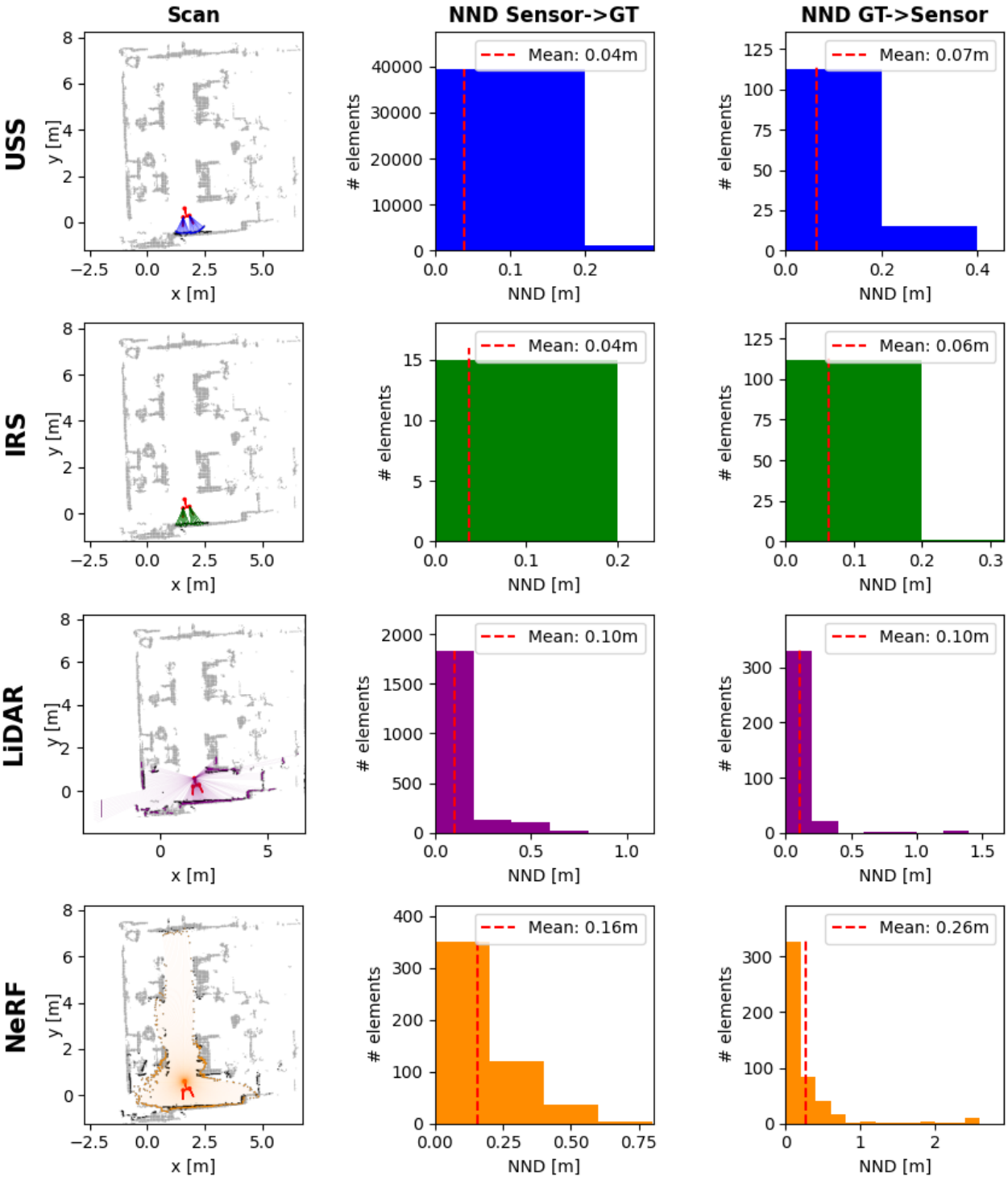


Figure C.7: *Office* example map (test point 44): In the first row, the global map is in grey and the GT scan is in black. Columns 2 and 3 show histograms of the number of measurements as a function of the NND for the accuracy and coverage respectively. The bin size is 0.2 m. The GT is limited to the FoV ($GT[FoV] \rightarrow Sensor$).

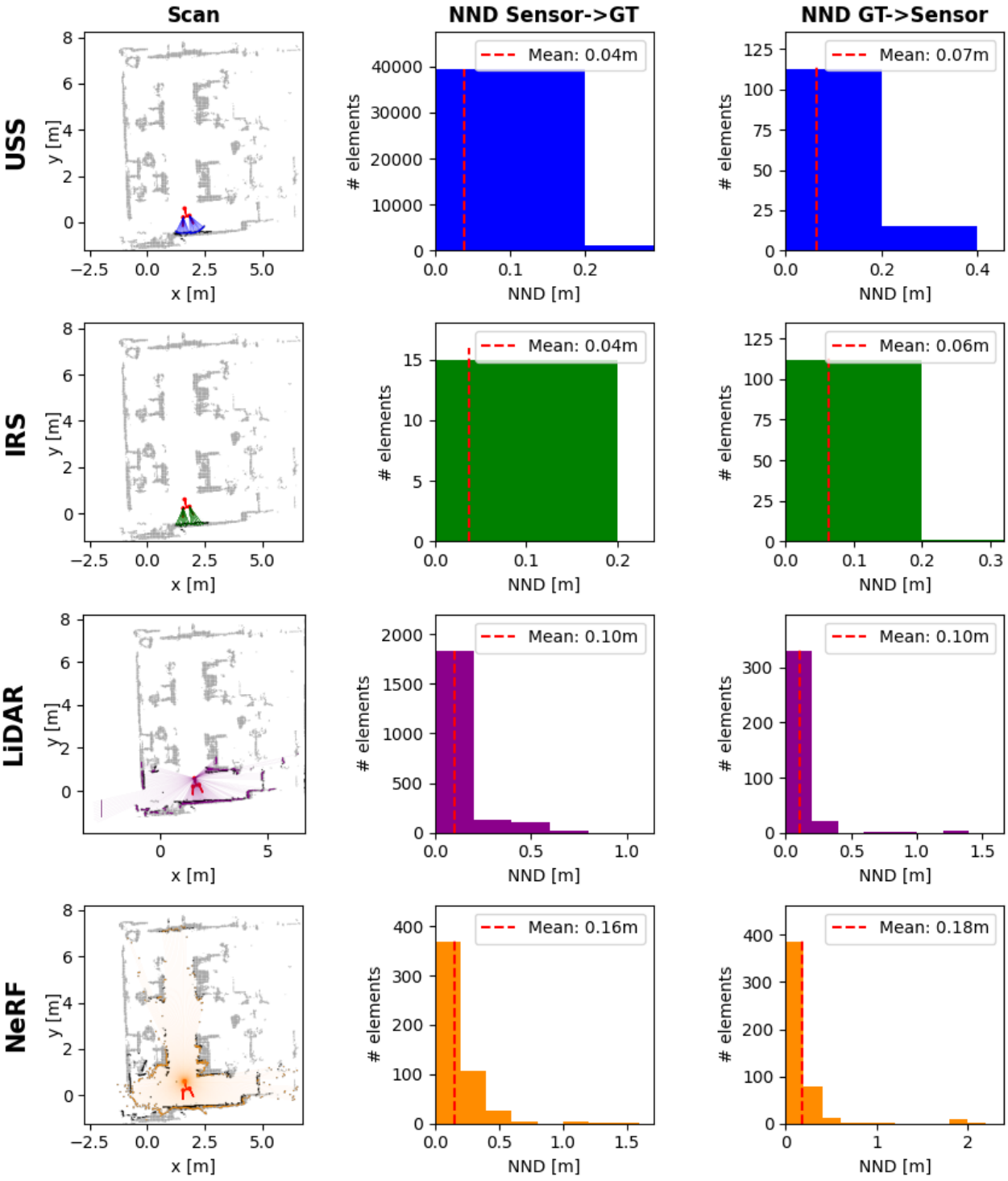


Figure C.8: *Office* example map (test point 44) using RGB-D camera: In the first row, the global map is in grey and the GT scan is in black. Columns 2 and 3 show histograms of the number of measurements as a function of the NND for the accuracy and coverage respectively. The bin size is 0.2 m. The GT is limited to the FoV ($GT[FoV] \rightarrow Sensor$).

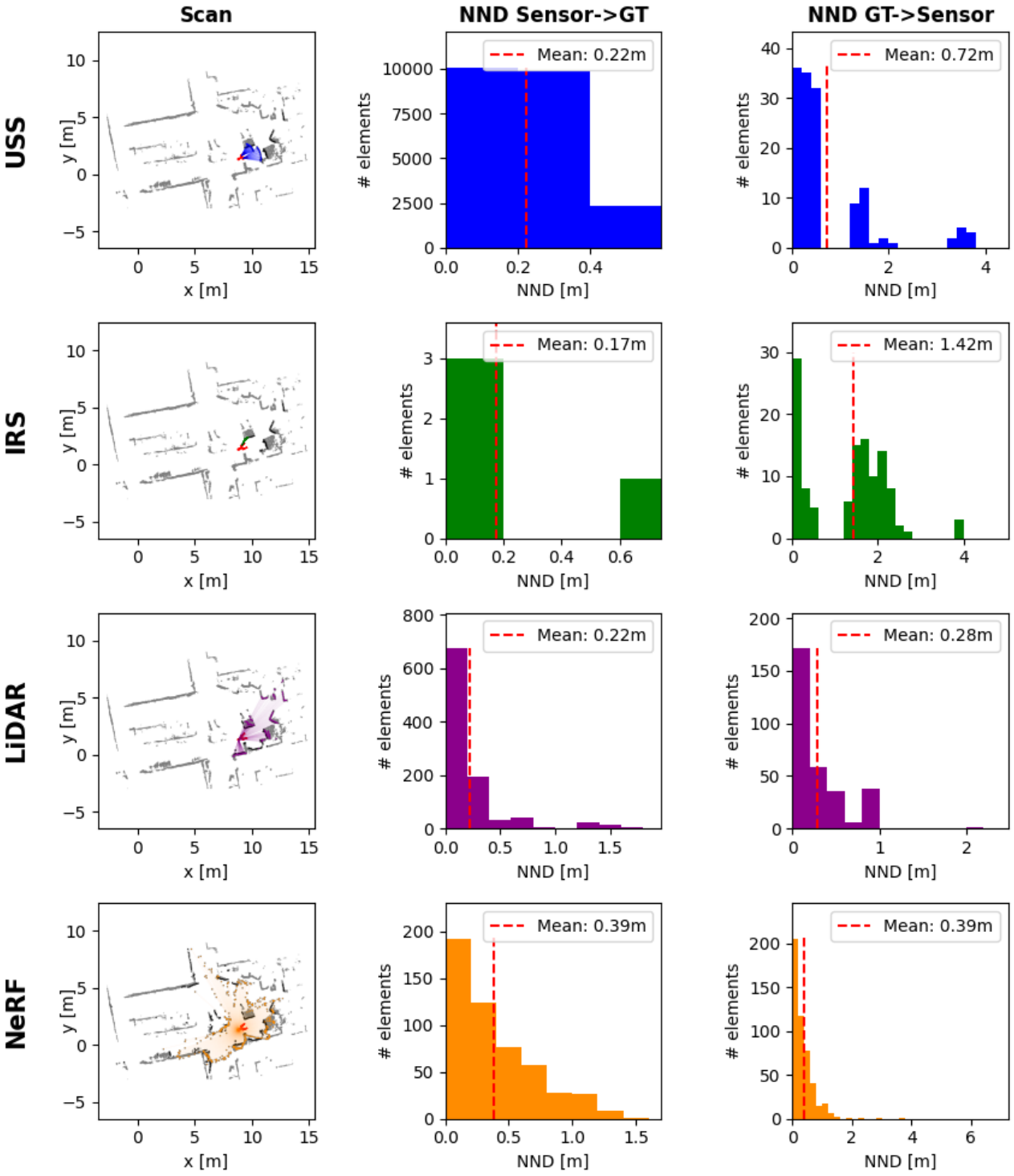


Figure C.9: *Common area* example map (test point 11): In the first row, the global map is in grey and the GT scan is in black. Columns 2 and 3 show histograms of the number of measurements as a function of the NND for the accuracy and coverage respectively. The bin size is 0.2 m. The GT is limited to the FoV ($GT[FoV] \rightarrow Sensor$).

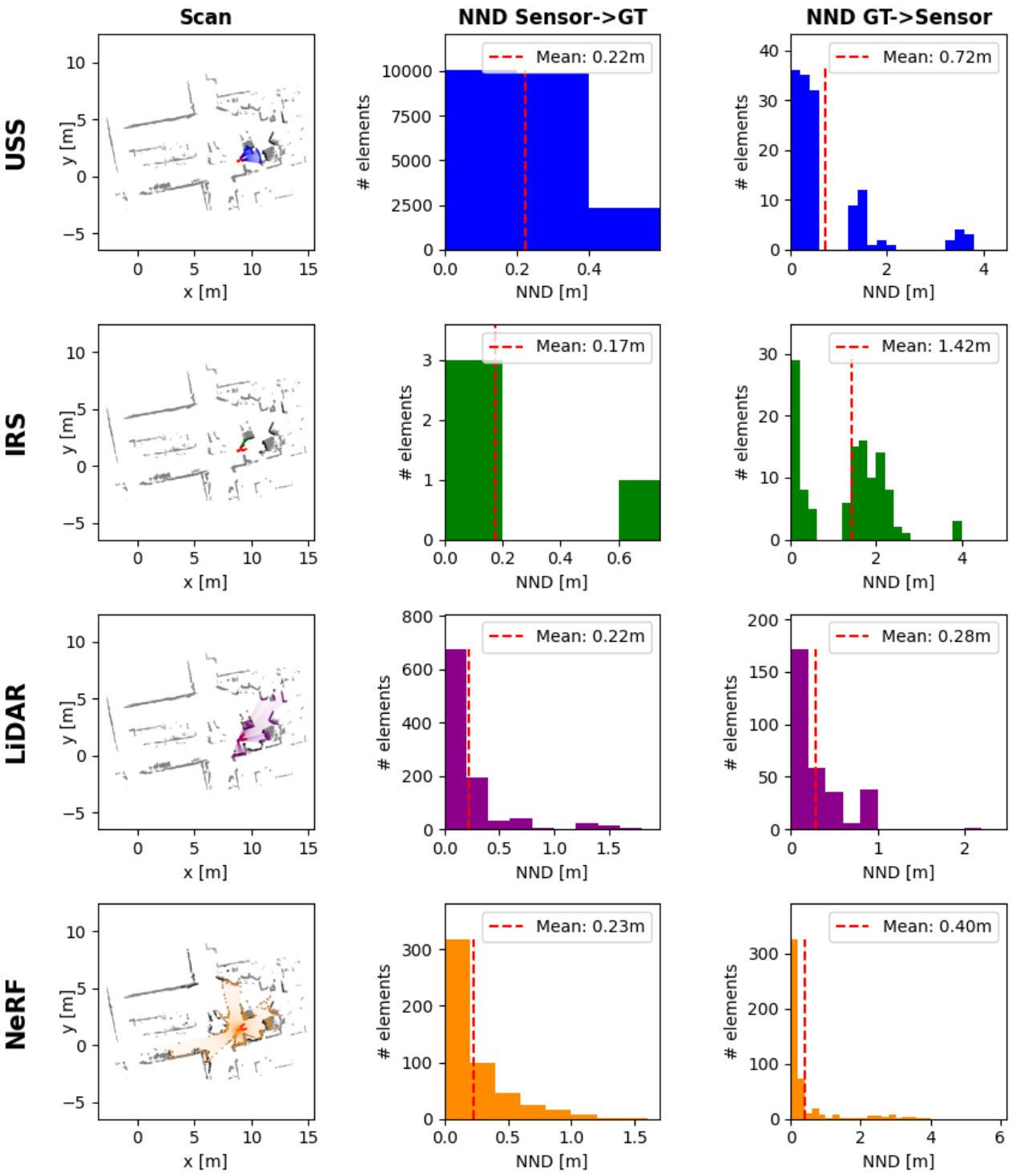


Figure C.10: *Common area* example map (test point 11) using RGB-D camera: In the first row, the global map is in grey and the GT scan is in black. Columns 2 and 3 show histograms of the number of measurements as a function of the NND for the accuracy and coverage respectively. The bin size is 0.2 m. The GT is limited to the FoV ($GT[FoV] \rightarrow Sensor$).

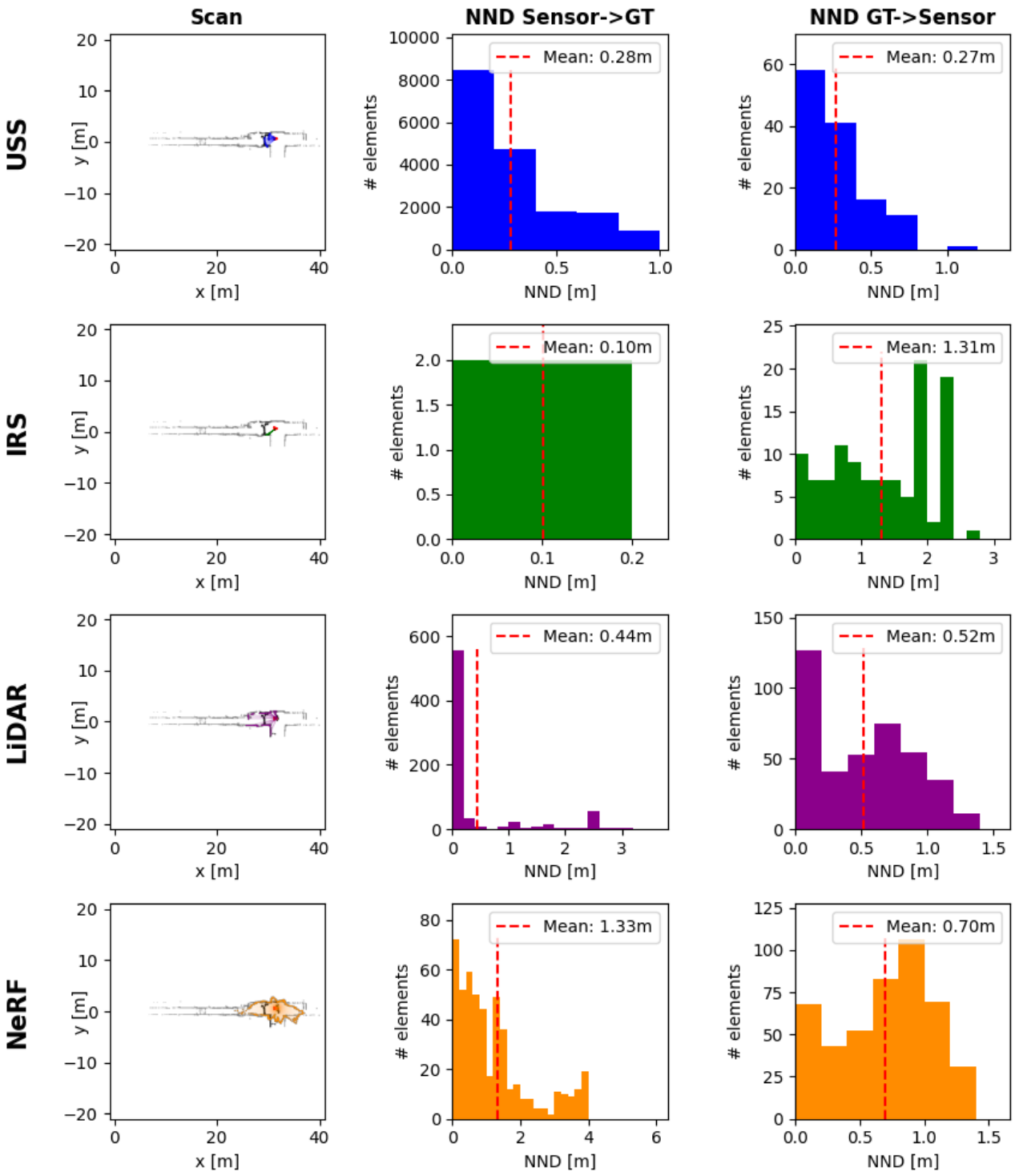


Figure C.11: *Corridor* example map (test point 37): In the first row, the global map is in grey and the GT scan is in black. Columns 2 and 3 show histograms of the number of measurements as a function of the NND for the accuracy and coverage respectively. The bin size is 0.2 m. The GT is limited to the FoV ($GT[FoV] \rightarrow Sensor$).

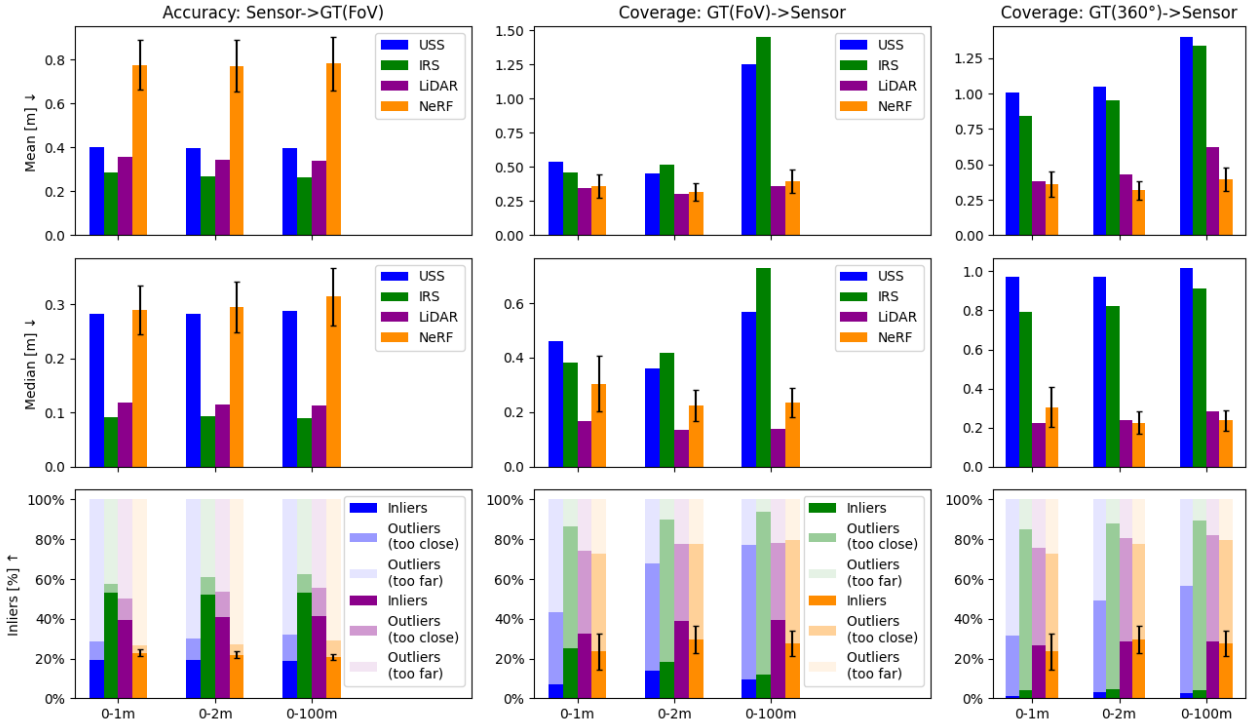
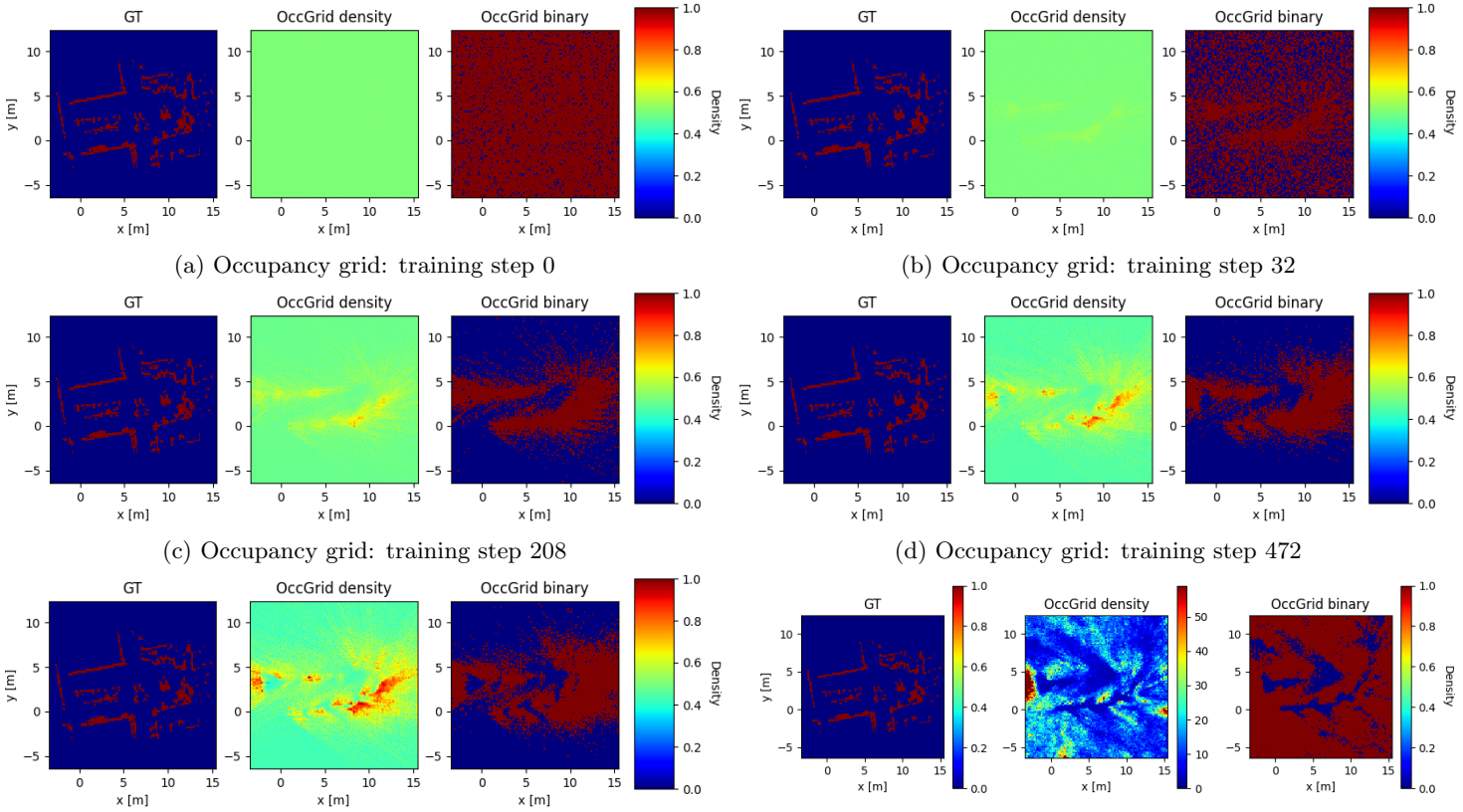


Figure C.12: *Corridor* NND: The first column describes the accuracy ($Sensor \rightarrow GT$) and the second and third ones the coverage limited to the FoV ($GT[FoV] \rightarrow Sensor$) or using all-around GT ($GT[360°] \rightarrow Sensor$). The rows show the mean NND, the median NND and the inlier ($NND < 10cm$) percentage. Outlier predictions are either *too close* or *too far* away relative to the robot. Each metric is calculated for three zones defined by the GT depth. *VIRUS-NeRF* is evaluated for 10 runs and the error bar indicates the standard deviation.

C.3.2 Occupancy Grids



(a) Occupancy grid: training step 0

(b) Occupancy grid: training step 32

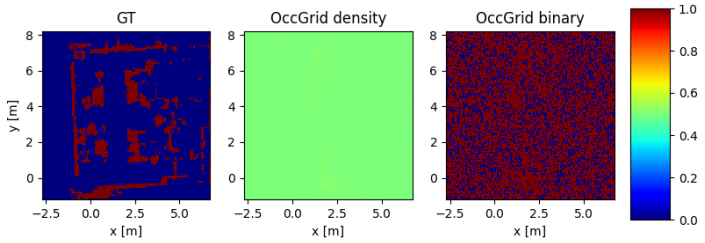
(c) Occupancy grid: training step 208

(d) Occupancy grid: training step 472

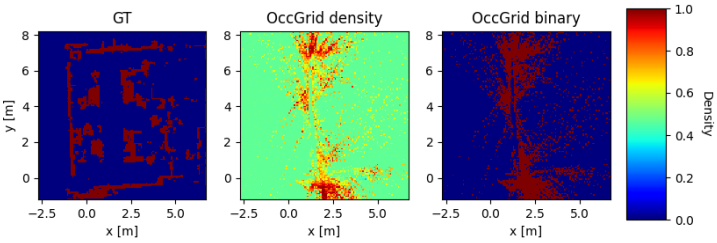
(e) Occupancy grid: training step 792 (final)

(f) *Instant-NGP* grid: training step 480 (final)

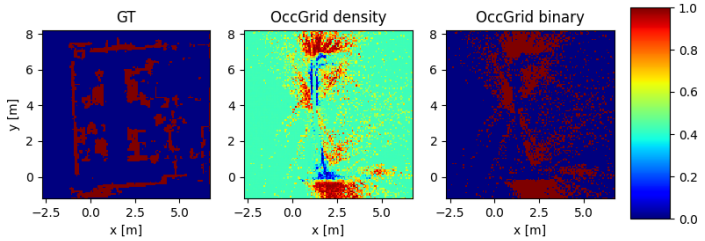
Figure C.13: *Common room* occupancy grids for several training steps: The grids are generated by *VIRUS-NeRF* except for subplot f which shows the final *Instant-NGP* grid.



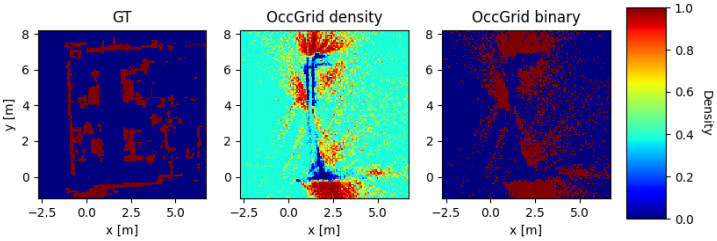
(a) Occupancy grid: training step 0



(b) Occupancy grid: training step 210

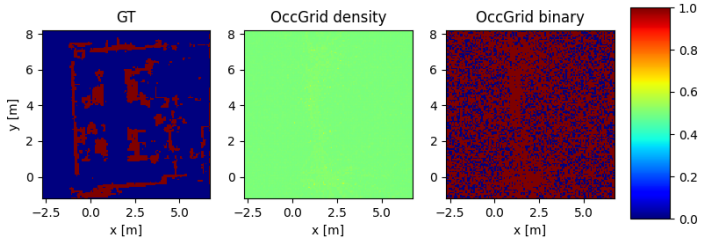


(c) Occupancy grid: training step 483

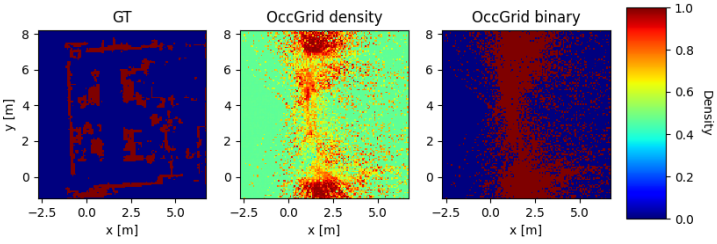


(d) Occupancy grid: training step 798 (final)

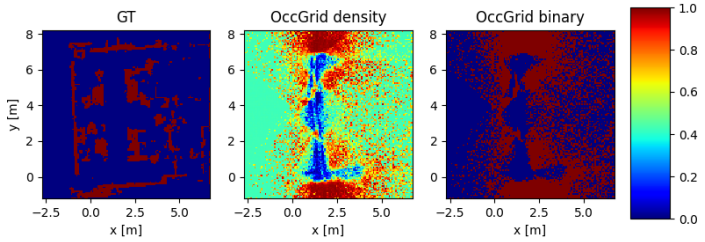
Figure C.14: *Office* occupancy grids for several training steps: The grids are generated by *VIRUS-NeRF* without using USSs (based on IRSs and cameras).



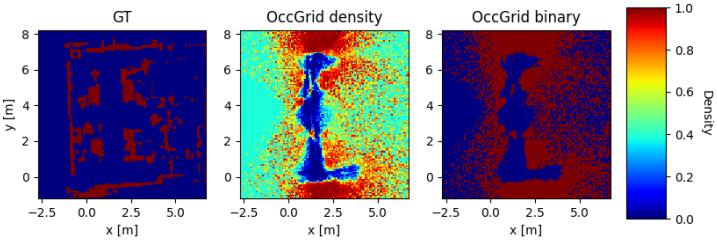
(a) Occupancy grid: training step 0



(b) Occupancy grid: training step 210



(c) Occupancy grid: training step 483



(d) Occupancy grid: training step 798 (final)

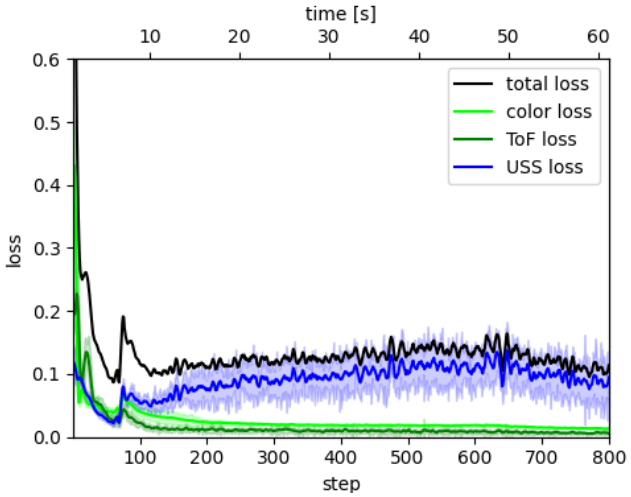
Figure C.15: *Office* occupancy grids for several training steps: The grids are generated by *VIRUS-NeRF* without using IRSs (based on USSs and cameras).

C.3.3 Ablation Study

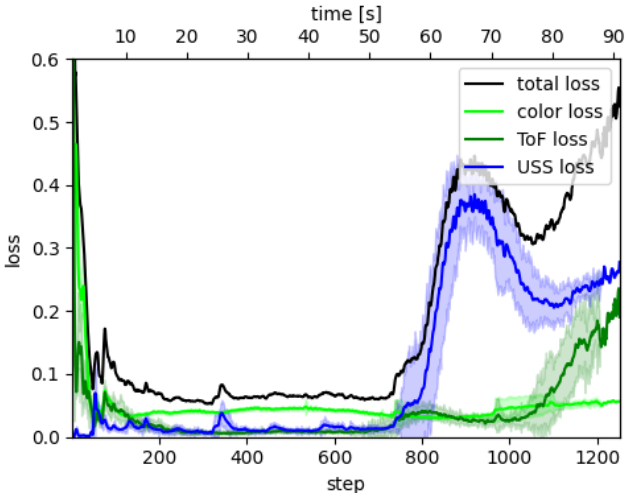
		Mean [m] ↓		Median [m] ↓		Inliers [%] ↑	
		Sensor →GT	GT(360°) →Sensor	Sensor →GT	GT(360°) →Sensor	Sensor →GT	GT(360°) →Sensor
Camera	mean	0.277	0.476	0.233	0.279	0.231	0.215
	std	*0.013*	*0.077*	*0.012*	*0.025*	*0.020*	*0.022*
Camera & USS	mean	0.262	0.349	0.228	0.246	0.212	0.176
	std	*0.027*	*0.014*	*0.027*	*0.026*	*0.042*	*0.039*
Camera & IRS	mean	0.23	0.374	0.16	0.121	0.415	0.46
	std	*0.008*	*0.021*	*0.016*	*0.013*	*0.016*	*0.018*
RGB-D	mean	0.154	0.139	0.079	0.074	0.575	0.633
	std	*0.006*	*0.006*	*0.006*	*0.004*	*0.022*	*0.017*
NGP Grid	mean	0.164	0.214	0.101	0.098	0.497	0.506
	std	*0.005*	*0.006*	*0.002*	*0.002*	*0.007*	*0.008*
Poses not optimized	mean	0.168	0.238	0.088	0.081	0.531	0.554
	std	*0.005*	*0.011*	*0.004*	*0.003*	*0.010*	*0.011*
Params not optimized	mean	0.172	0.259	0.113	0.115	0.467	0.459
	std	*0.004*	*0.011*	*0.006*	*0.004*	*0.015*	*0.011*
Optimized Particle 2	mean	0.148	0.237	0.09	0.09	0.528	0.531
	std	*0.005*	*0.014*	*0.006*	*0.004*	*0.016*	*0.014*

Table C.1: Ablation Study *Office*: The NND is calculated for zone 3 ($0 - 100\,\text{m}$) and averaged over 10 runs showing the standard deviation. All results are generally produced by *VIRUS-NeRF* with the optimized poses and hyper-parameters (last row). The first four rows show different sensor constellations by removing USSs and IRSs or by using depth cameras. The fifth row is *VIRUS-NeRF* but using the occupancy grid of *Instant-NGP*. In rows 6 and 7, the poses or the hyper-parameters are not optimized respectively. Dark orange, light orange and yellow indicate the best, second and third-best results respectively.

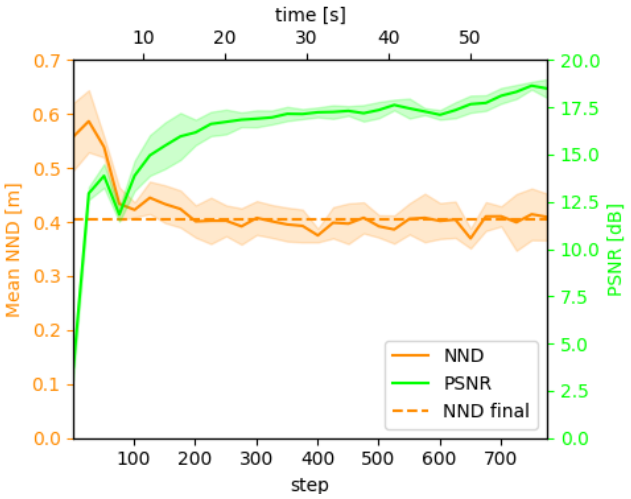
C.4 Training Speed



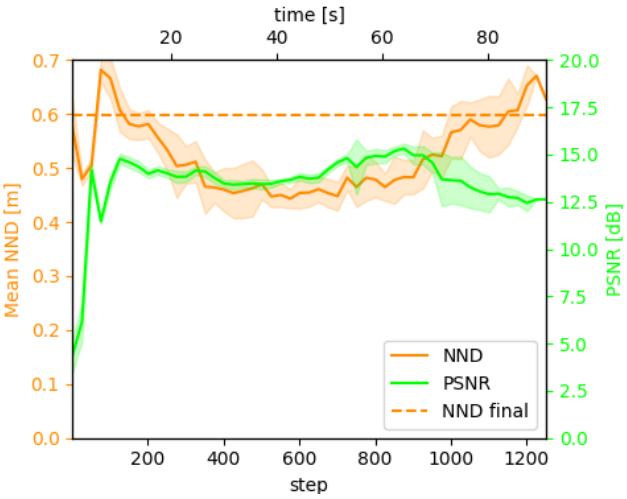
(a) Offline training losses



(b) Online training losses



(c) Offline training metrics



(d) Online training metrics

Figure C.16: *Common area* training curves: The coloured area indicates the standard deviation over 10 runs. The mean losses are smoothed with a Savitzky-Golay filter (order 3, window size 10). The NND is the accuracy ($Sensor \rightarrow GT$) in zone 3 ($0 - 100\,\mathrm{m}$).