UNIVERSITÀ DEGLI STUDI DI PADOVA

DEPARTMENT OF INFORMATION ENGINEERING (DEI)

Doctoral Thesis in INFORMATION ENGINEERING

GOING FROM iToF TO dToF: METHODS FOR MPI CORRECTION AND TRANSIENT RECONSTRUCTION

Supervisor
PIETRO ZANUTTIGH
UNIVERSITÀ DI PADOVA

Co-supervisor
GIANLUCA AGRESTI
SONY R&D CENTER STUTTGART LABORATORY 1

Co-supervisor
HENRIK SCHÄFER
SONY R&D CENTER STUTTGART LABORATORY 1

Ph.D. Candidate
ADRIANO SIMONETTO

Ai miei genitori per tutto il supporto e la pazienza

Abstract

Depth data contains vital information for understanding the geometry of objects and placing them in the 3D world. To this aim, several sensors have been developed to retrieve depth images, each with its own strength and weaknesses. In this thesis we will cover Time-of-Flight cameras (ToF), a particular kind of sensor which computes the distance information based on the time of flight of a light impulse.

We will mainly focus on indirect ToF cameras, which are commercially available devices working at interactive frame rates with until 1 Megapixel resolution. These devices are affected by Multi-Path Interference (MPI), a key challenge causing distortion in the depth estimation process. Common data-driven approaches tend to focus on a direct estimation of the output depth values, ignoring the underlying transient propagation of the light in the scene. In this Ph.D. thesis instead, we propose some very compact deep learning architectures for transient data estimation, the first one exploiting a two-peaks encoding of the transient, the second one leveraging on the direct-global subdivision of transient information.

Afterwards, we will instead deal with ToF sensors in a power-constrained environment, such as that of mobile devices. In this setting, we will propose a quantized neural network for depth completion, which reaches competitive performance while keeping its size limited. Finally, we switch to hyperspectral images and see a work on learning architectures for spectrum reconstruction with limited amounts of ground truth data.

Sommario

L'informazione relativa alla profondità è vitale per comprendere la geometria degli oggetti e inserirli in uno spazio 3D. Per questa ragione sono stati introdotti numerosi sensori che producono immagini di profondità, ciascuno con i suoi punti di forza e di debolezza. In questa tesi ci concentreremo su telecamere a tempo di volo, un tipo particolare di sensore che calcola l'informazione relativa alla distanza basandosi sul tempo di volo di un impulso luminoso.

Ci concentreremo in primo luogo su telecamere a tempo di volo indirette, che sono dispositivi in commercio con frequenze di cattura sufficientemente elevate e risoluzione sino a 1 Megapixel. Le immagini prodotte da questi apparecchi soffrono di una distorsione chiamata Multi-Path Interference (MPI), un problema di primo piano legato a questa tecnologia. Gli approcci più comuni per risolvere questa distorsione cercano di stimare direttamente i valori corretti di profondità, ignorando la propagazione temporale della luce nella scena. In questa tesi di dottorato invece, proponiamo alcuni metodi molto compatti basati su deep learning per la stima del comportamento transitorio della luce; il primo sfrutta una codifica a due picchi di questa funzione, mentre il secondo introduce una suddivisione tra informazione diretta e globale.

In seguito, considereremo sensori a tempo di volo in condizioni dove la potenza a disposizione è limitata, come ad esempio per dispositivi mobile. In questo caso, proponiamo una rete neurale quantizzata per depth completion, che raggiunge prestazioni competitive mantenendo comunque dimensioni ridotte. Infine, cambieremo tema introducendo le immagini iperspettrali e vedremo un lavoro sulla ricostruzione di informazione iperspettrale nel caso in cui i dati per l'allenamento scarseggino.

Contents

List of Figures

List of Tables

1 Introduction

The demand for more accurate and reliable range imaging devices has seen a constant rise over the years. Their applications are widespread ranging from autonomous driving [8, 9] to augmented reality [10], 3D reconstruction [11, 12] and even landing on planetary bodies [13]. The working principles are different for the various types of sensors, but the main objective remains the same: retrieving the distance information between the camera and the target object. Some of the most common technologies are stereo imaging [14], where the depth information is retrieved from a couple of RGB cameras at a fixed distance, Time-of-Flight (ToF) based devices [15], e.g. LIDARs [16] or matrix ToF sensors, and structured light scanners [17], that rely on light patterns.

In this thesis we will focus our attention on ToF based technologies, more specifically on indirect Time-of-Flight (iToF) cameras. A direct Time-of-Flight (dToF) device sends an impulse of light towards the scene, measures the travel time of the impulse and computes the depth information from that. An iToF camera instead sends a modulated light signal and correlates the reflected signal with the sensor modulation signal; from these measures the distance is retrieved. iToF-based cameras are quite accurate, have a good spatial resolution and are nowadays sold at consumer level in some of the most recent mobile phones [18]. This technology however comes with its disadvantages, with the top spot taken by Multi-Path Interference (MPI). This error source is intrinsic to the technology and produces an overestimation of the depth strictly linked to the scene geometry which has been widely studied in the literature [19, 20, 21, 22]. Another downside is linked to energy consumption; while common RGB cameras can rely on natural illumination, ToF sensors need an active illuminator, which can have a toll on the limited power budget of a mobile device. In this thesis, we will investigate the limitations of iToF devices and propose some techniques to alleviate them while at the same time keeping a low network complexity. A major focus will be devoted to the tasks of MPI correction and transient reconstruction, two problems that, as we will see,

are very closely related.

The final part of this thesis will cover instead hypespectral sensors. As we will see, these devices have issues related to their cost and the acquisition/processing time. We will cover the topic of spectrum reconstruction and introduce a few techniques for training deep learning architectures with limited ground truth.

The results and methods included in this thesis comprehend the work done during the three years of my Ph.D.. My studies have been funded by Sony Europe B.V., and I also spent one year at their research centre in Stuttgart, where I had the occasion of working together with experts on the topic and with the sensors themselves.

The rest of this thesis will be structured as follows. Chapter 2 will provide a thorough description of Time-of-Flight sensors, starting from their operating principle, and concluding with the main error sources. Chapter 3 will define the tasks of Multi-Path Interference correction and transient reconstruction and the relationship between the two. In Chapter 4 we will instead see the employed datasets, both iToF and transient, with a focus on the *Walls* dataset which was developed during my first year of Ph.D.. After that, in Chapter 5 we will explain the approaches we propose for the two mentioned tasks. We will start with [22], a deep learning approach exploiting a rough encoding of transient information. Afterwards, we will introduce [23], a more refined version of [22], which reaches state of the art performance while keeping and extremely low network complexity. At the end of the chapter we will show a thorough comparison with the best performing approaches from the literature and draw our conclusions for this part. In Chapter 6 instead, we will talk about depth completion, we will introduce the surrounding literature and conclude the chapter by describing the datasets for the task. The main approach [24] and related results for depth completion will be instead introduced in Chapter 7. Then, Chapter 8 will be devoted to the approach [25] that we propose for the task of spectrum reconstruction. Finally, in Chapter 9 we will draw our conclusions.

2 Time-of-Flight Sensors

In this chapter we will give an introduction to Time-of-Flight sensors, a widely used technology for retrieving depth information. We will start with direct Time-of-Flight sensors, and then continue with indirect Time-of-Flight sensors, which are the main focus of this Ph.D. thesis. We will then describe the limitations of iToF devices, with a particular interest for Multi-Path Interference, an error source inherently linked to their operating principle.

2.1 Direct Time-of-Flight

The operating principle behind direct Time-of-Flight sensors is quite simple. Since the speed of light c is fixed, we can think of sending a very narrow impulse of light towards a scene, compute the round trip time Δt at the sensor side, and then from it retrieve the depth d with the following relation:

$$d = \frac{c\Delta t}{2}, \tag{2.1}$$

where the $\frac{1}{2}$ factor is due the fact that light travelled to the object and came back.

While at a first glance this may appear straightforward, it poses several complex challenges from a hardware perspective linked to the high value of c. To give a practical example, if we aim for a depth sensitivity of $d_{sen} = 1\,\mathrm{cm}$, we are going to need a sensor with a temporal sensitivity of $t_{sen} = \frac{2d_{sen}}{c} = \frac{2 \cdot 0.01\mathrm{m}}{3 \cdot 10^8 \frac{\mathrm{m}}{\mathrm{s}^2}} = 6.67 \cdot 10^{-11}\,\mathrm{s}$ which are $66.7\,\mathrm{ps}$. In practice, it was only at the beginning of the 2000s that these tight requirements could be met for the acquisition of a scene with a single pulse, with the use of Single-Photon Avalanche Diodes (SPAD) arrays [28, 29]. A SPAD array consists of a set of diodes working in Geiger mode, a state where the arrival of a single photon can cause an avalanche effect [30]. The avalanche happens in an extremely narrow time-frame and allows to precisely trace the original time of arrival of the incoming photon.

DToF cameras based on SPAD arrays can reach a time sensitivity of 12 ps [31] and have recently crossed the 1 Megapixel resolution [32]. One of the main limitations of SPAD arrays is their limited fill factor, which is the ratio between the light sensitive area of a pixel and the total size of the pixel. In practice, SPAD arrays need a high degree of parallelism and timing circuitry can cover a good chunk of the pixel space [33]. Another phenomenon constraining the fill factor is crosstalk between pixels, a phenomenon where an avalanche event triggered at a pixel, spreads to neighbouring ones causing false detections [34].

2.2 Indirect Time-of-Flight

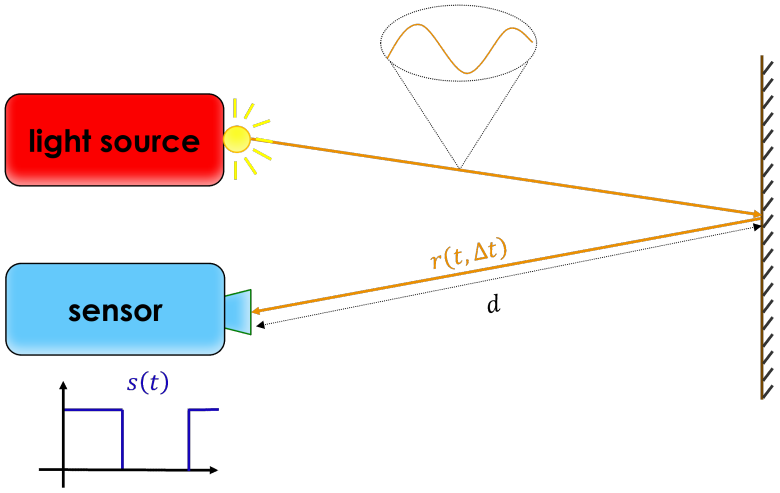


Figure 2.1: Common setup for indirect ToF cameras

Indirect Time-of-Flight sensors follow the same high level idea, but put it in practice in a different manner. An iToF camera is composed of a light source emitting a modulated signal and an active sensor. Figure 2.1 shows an example of a common setup. Usually, the modulated light source of the emitter is a sine wave in the frequency of $10 - 100$ MHz, while the sensor sensitivity function consists of a square wave at the same frequency. This combination of sine and square wave at the same modulation frequency is particularly useful from a theoretical perspective as it allows for a closed form solution for the reconstruction of the distance. In the remainder of this section, first the operating principles of iToF cameras will be reviewed, and then the main noise sources with a focus on Multi-Path Interference will be described.

2.2.1 Operating Principle

The emitter of an iToF camera sends towards the scene a sinusoidal signal $i(t)$ of the form:

$$i(t) = i_0 + a\sin(2\pi f_m t), \tag{2.2}$$

where i_0 is the intensity of the signal, a its amplitude and f_m the employed modulation frequency. The sensor sensitivity function $s(t)$ is expressed as the Heavyside step function of the sinusoidal signal $i(t) - i_0$:

$$s(t) = H(\sin(2\pi f_m t)). \tag{2.3}$$

The sensor will then receive the reflected signal $r(t)$ which consists of a delayed version of $i(t)$ scaled by the optical response of the scene ρ:

$$r(t) = \rho \cdot i(t - \Delta t) = \rho \cdot (i_0 + a\sin(2\pi f_m(t - \Delta t))) \tag{2.4}$$

with Δt the time delay. The same delay can also be seen as a phase shift $\varphi = 2\pi f_{mod}\Delta t$ in the frequency domain. At the sensor side, the returning signal and the sensor function are integrated, leading to the camera measurements m_θ:

$$m_\theta = \int_0^{T_{int}} r(t)s\Big(t + \frac{\theta}{2\pi f_m}\Big)\mathrm{d}t = I + A \cdot \cos(\varphi + \theta), \tag{2.5}$$

with T_{int} the integration time and θ an internal phase displacement applied to the sensor sensitivity. The quantities I, A and φ, respectively intensity, amplitude and phase delay of the recovered ToF signal, are three unknowns which we can estimate by sampling equation (2.5) a total of 4 times using 4 different values of θ. Commonly, the chosen values are $\theta \in [0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}]$ which give us the following relations:

$$A = \frac{1}{2}\sqrt{(m_0 - m_\pi)^2 + (m_{\frac{3\pi}{2}} - m_{\frac{\pi}{2}})^2}, \tag{2.6}$$

$$\varphi = \arctan2\left(m_{\frac{3\pi}{2}} - m_{\frac{\pi}{2}}, m_0 - m_\pi\right), \tag{2.7}$$

$$I = \frac{m_0 + m_{\frac{\pi}{2}} + m_\pi + m_{\frac{3\pi}{2}}}{4}. \tag{2.8}$$

where the arctan2 is a variation of the arctan which has output values in the interval $(-\pi, \pi]$. Finally, from the value of φ we can directly retrieve the depth value d as follows:

$$d = \frac{c \cdot \varphi}{4\pi f_m} \tag{2.9}$$

Moreover, following the work of Gupta er al. [35], it is convenient to represent the camera measurements in phasor notation. In practice, we can see $m_0 - m_\pi$ as the real component of a phasor and $m_{\frac{3\pi}{2}} - m_{\frac{\pi}{2}}$ as its imaginary part. This leads to the following expression:

$$\mathbf{v} = Ae^{i\varphi} = Ae^{i2\pi f_m \Delta t} \in \mathbb{C}, \tag{2.10}$$

where A and φ are respectively the amplitude and phase of the original sinusoidal function.

Note that since the whole theory behind this technology is based on periodic signals, there is an inherent limitation to its maximum range. In practice, the *ambiguity range* of an iToF camera, consists of

$$d_{max} = \frac{c \cdot 2\pi}{4\pi \cdot f_m}. \tag{2.11}$$

For example, in the case of a modulation frequency of $10\mathrm{MHz}$, the ambiguity range amounts to $15\mathrm{m}$. Even though there are ways to extend the ambiguity range when working with multiple frequencies [36], iToF sensors are mostly suited for short-range, indoor environments.

2.2.2 Multi-Path Interference

We have considered the case in which the ToF signal is reflected only once inside the scene. However, in real scenarios it is highly likely for the light to be reflected multiple times, causing numerous light rays to arrive at the same pixel. This effect is called Multi-Path Interference. A common case of MPI can be seen in Figure 2.2, where the correct distance of the measured pixel (in orange) will be overestimated due to light bouncing on a second surface (in red). In this case, it is possible to generalize the above description by assuming that the resulting ToF signal is the summation of the different interfering signals, each one described as a phasor. Recall that sinusoidal signals with the same frequency, as well as their phasor representation, are closed under summation

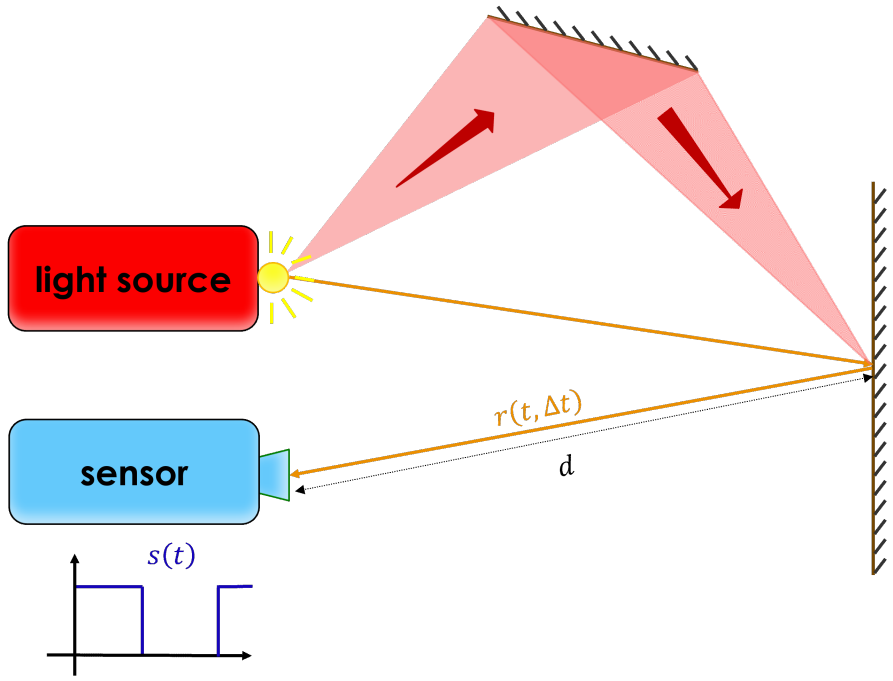


Figure 2.2: Image acquisition affected by Multi-Path Interference. The path in orange is the one delivering the correct depth value. The interfering rays (in red) bounce on a second surface, then hit the main surface and come back to the camera. Since they follow a longer trajectory, the final measurement will be overestimated.

[35]. As a consequence, a ToF measurement originated by MPI can be described as

$$v = \int_{t_{min}}^{t_{max}} x(t) e^{i2\pi f_m t} \mathrm{d}t, \tag{2.12}$$

where t_{max} is the maximum time of flight of the considered interfering rays and $x(t)$ is a function describing the strength of the interfering rays coming back to the sensor at time t. This function describes the behaviour of light through time and for this reason, as we will see later on, it is known as transient image in its discrete version. In Figure 2.3 we can see an example of a transient pixel, where the strong first reflection indicating the correct depth value corresponds to the first peak, while all the remaining components correspond to the secondary light bounces. The MPI phenomenon described above is a non-zero mean error in ToF depth measurements, usually leading to an overestimation of depth. A key aspect of MPI distortion is its dependency on the geometry of the considered scene, which heavily influences the light transient function $x(t)$. An example can be seen in Figure 2.4, where we can see the error due to MPI on

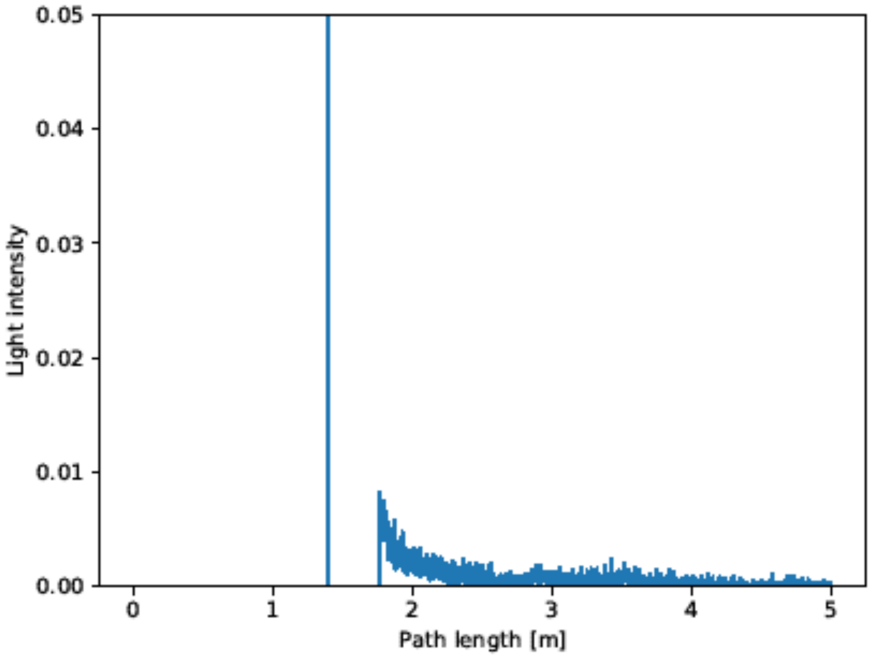


Figure 2.3: A transient pixel showing the incoming light radiance at each time step.

a scene composed of three walls. The green parts of the image correspond to a correct depth estimation, while the ones in red mean an overestimation due to MPI. As we can see, this kind of distortion is particularly strong close to the corners between walls, and gradually decrease in intensity farther away from them, due to the inverse relationship between light intensity and path length. Another interesting property of MPI is the fact that the error pattern changes with the modulation frequency and in particular, the higher the frequency, the lower the error due to MPI. The reason behind this, is linked to the integral from Equation (2.12), where we are correlating the transient pixel in Figure 2.3 with a sinusoidal function. If the modulation frequency of the periodic function is high the components of the secondary light bounces tend to partially cancel out as they are spread through the time dimension leading to a lower depth error. In the case instead of a lower modulation frequency these tend to be all weighted more or less in the same manner and the end result is a higher MPI-related distortion. In general however, it is not possible to simply choose a high value for f_m to reduce as much as possible the MPI since, as we have seen in Equation 2.11, modulation frequency

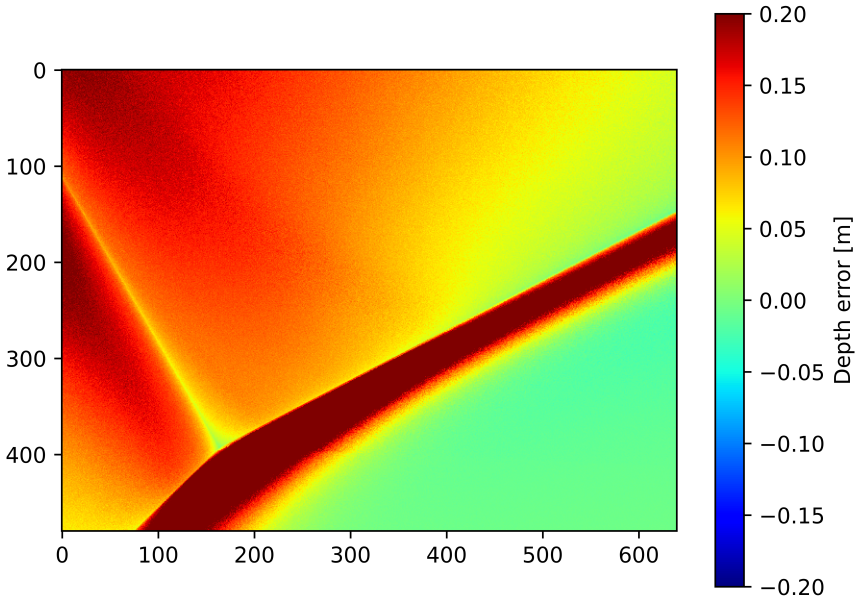


Figure 2.4: Error caused by MPI. Values correctly estimated are in green, while the red parts show an overestimation of the correct depth.

and ambiguity range are inversely proportional. For example, setting f_m to 200 MHz would indeed yield a very low MPI error, but would also reduce the maximum range to $0.75\mathrm{m}$. In conclusion, while the problem of MPI correction has been widely studied in the literature [20, 37, 38, 39], it is still challenging.

2.2.3 Photon Shot Noise

The accuracy of the depth measurements is strictly related to the intensity of the reflected light. In scenarios where the returning illumination is quite high, the measurement is going to be quite accurate, while the opposite is true for poor light conditions. This noise source, commonly called *photon shot noise* or, in short, *shot noise*, is associated with the particle nature of light and arises from the photon to electron conversion. It can be modelled as a Poisson random variable whose mean (μ_p) and variance (σ_p^2) are both equal to the number of incoming photons. For this reason, the SNR ($\frac{\mu_p}{\sigma_p}$) is going to increase with the square root of the number of arriving photons.

2.2.4 System Noise

System noise comprises the noise directly related to the electronics involved in the measurement process. In particular, it is comprised of reset noise, as the reset values of pixels are affected by small variations, and of read noise, which is generated by electronics as the charge present in the pixels is transferred to the camera. Note that differently from shot noise, system noise is pure noise, completely independent of the measurement value.

2.2.5 Wiggling Error

All the theory behind iToF cameras introduced in Section 2.2.1 requires an ideal sinusoidal signal at the emitter side, and an ideal square wave at the sensor side. In practice however, this is not the case for common iToF cameras, where instead signal $i(t)$ consists of something in between a sinusoid and a rectangular wave. This means that in this non-ideal scenario sampling 4 values would not be enough for a complete reconstruction of the signal and would lead to aliasing effects. The result is the so called cyclic or wiggling error, a distortion easily noticeable over flat surfaces, where instead the measured signal shows some periodic bumps. The most effective and employed solution is to use a *look-up* table approach as proposed by Lindner et al. [40].

3 MPI Correction and Transient Reconstruction

In this chapter, we will introduce the two main topics of this Ph.D. work, which are MPI correction and transient reconstruction, and then proceed to illustrate the surrounding literature for both topics.

3.1 Multi-Path Interference Correction

As introduced in Chapter 2, one of the main limitations of iToF cameras is Multi-Path Interference. This phenomenon, which causes an overestimation of the depth due to secondary bounces of light, is highly challenging as it depends on the structure of the scene and on the properties of reflecting materials. The objective of MPI correction is to retrieve clean depth measurements given single or multi-frequency iToF information, either in their raw measurements form, or in the form of depth amplitude images.

3.2 Transient Reconstruction

The task of transient reconstruction is strongly intertwined with that of MPI correction, which can be seen as a by-product of it. In practice, given some multi-frequency iToF acquisitions, the objective is to retrieve the temporal information of light coming into the sensor. Referring to Equation (2.12), we have the iToF measurements v, and we want to invert the integral to obtain $x(t)$.
Equation (2.12) has an integral formulation, which however is not practical as all the measurements we are dealing with are discrete. For this reason, we consider the discrete version of this equation by sampling the time interval of integration into N time steps. This allows to rewrite the integral as a summation:

$$v = \sum_{j=1}^{N} x_i e^{i2\pi f_m t_j} \in \mathbb{C}, \tag{3.1}$$

which can be expressed as an inner product between two vectors

$$v = \begin{bmatrix} e^{i2\pi f_m t_0} & \dots & e^{i2\pi f_m t_{N-1}} \end{bmatrix} \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \end{bmatrix} = \mathbf{\Phi}' \boldsymbol{x}, \tag{3.2}$$

the first vecto $\mathbf{\Phi}' \in \mathbb{C}^{1\times N}$ corresponding to the measurement model and the second also known as *transient* or *backscattering* vector $\boldsymbol{x} \in \mathbb{R}^{N\times 1}$ to the scene impulse response. In Figure 3.1, we can see an example of a transient vector, where it is also possible to see the clear difference between the main reflection (in red), corresponding to the first peak, and all the other rays which bounced at least twice inside the scene (in green), corresponding to all the other non-zero bins. From now onwards, we will refer to the first peak as the *direct component*, and to all other reflections as the *global component*.

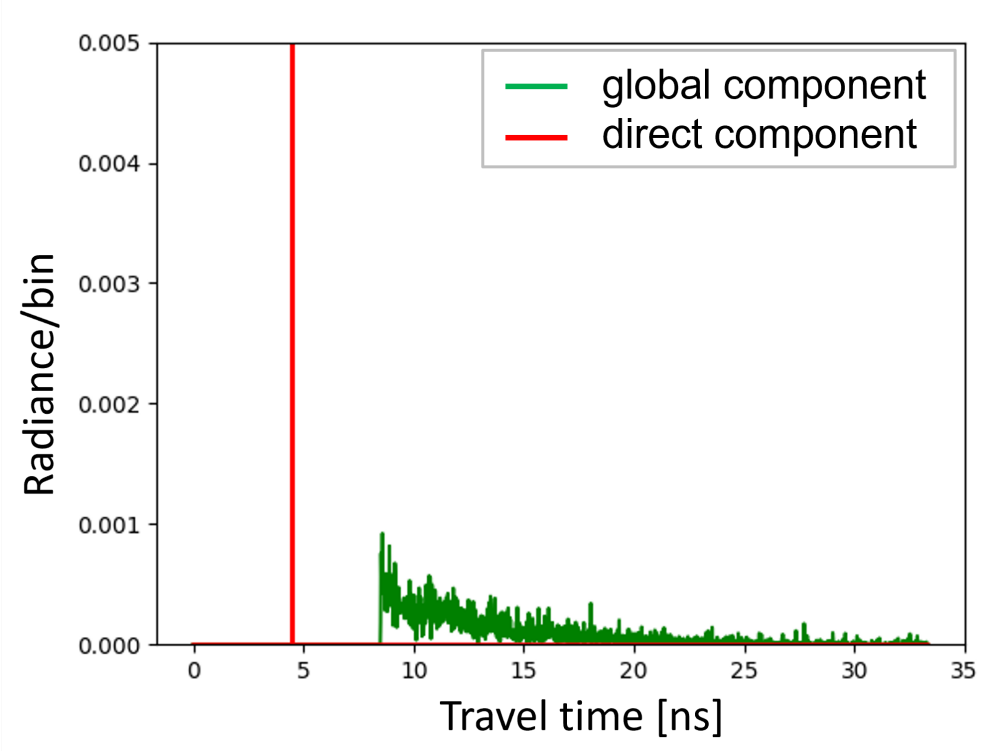


Figure 3.1: Transient vector where the direct and global components have been highlighted respectively in red and green.

Up to now, all the considerations have been made for a single frequency setup. It is useful to extend this to a set of acquisitions made at M different modulation frequencies. The reason behind this is that, as we mentioned in Chapter 2, the distortion pattern due to MPI changes with the frequency f_m, providing additional information; at the same time, this can also help getting a longer unambiguous range while keeping the same accuracy in the depth domain [36].
A quite straightforward generalization of Equation (3.2) to M input frequencies leads to the following expression:

$$\boldsymbol{v} = \begin{bmatrix} \boldsymbol{v_0} \\ \vdots \\ \boldsymbol{v_{M-1}} \end{bmatrix} = \begin{bmatrix} e^{i2\pi f_0 t_0} & \dots & e^{i2\pi f_0 t_{N-1}} \\ \vdots & \ddots & \vdots \\ e^{i2\pi f_{M-1} t_0} & \dots & e^{i2\pi f_{M-1} t_{N-1}} \end{bmatrix} \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \end{bmatrix} = \boldsymbol{\Phi x}, \tag{3.3}$$

where $\boldsymbol{v} \in \mathbb{C}^{M\times 1}$ is the stack of the raw camera measurements in the complex domain at different modulation frequencies, while $\boldsymbol{\Phi} \in \mathbb{C}^{M\times N}$.
Based on this discrete formulation, the problem of transient reconstruction is the following: given the raw measurements $\boldsymbol{v}$ and matrix $\boldsymbol{\Phi}$, we want to recover the transient vector $\boldsymbol{x}$. The hereby presented problem is heavily under-constrained as $N \gg M$, and has an infinite amount of solutions. A visualization of the problem can be seen in Figure 3.2.
Note, that while transient reconstruction can be seen as an iToF to dToF translation task, this is true only with some approximation. The transient that we aim to recover is an ideal version of the common output of a dToF camera, without any external noise source or limitation regarding the resolution in the time domain.

3.3 Related Works

The approaches that tackle MPI correction can be generally divided into two groups, single-frequency and multi-frequency ones. Those belonging to the first group such as [41, 42] and [43] exploit a reflection model together with the spatial information provided by the MPI-corrupted image for their solution. Jimenez et al [43] for example, proposed an iterative optimization algorithm based on the assumption that all scene surfaces are perfectly Lambertian. Early multi-frequency approaches show similar constraints. In [19], Freedman et al. introduced the relationship between iToF

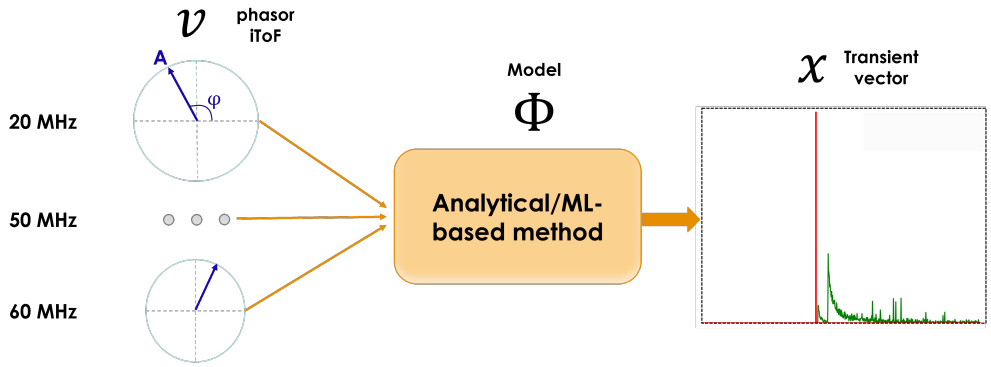


Figure 3.2: Transient reconstruction from iToF for 3 frequencies at 20, 50 and 60 MHz. Given multi-frequency iToF measurements v and matrix Φ, we want to reconstruct transient vector x

measurements and the transient behaviour of light extending the problem to the case of K interfering rays. They then proposed an algorithm for MPI correction treating it as an L_1 optimization problem. Bhandari et al. [44], adopted similar assumptions but offered instead a non-iterative solution using Vandermonde matrices.

The restrictions of these models and the unrealistic amount of input frequencies required for the solutions lead to a rapid rise in popularity of deep learning based approaches. Marco et al. [38] proposed an encoder-decoder architecture with a split training approach: the encoder was trained on unlabelled real data, and the decoder on the synthetic dataset they introduced. Su et al. [20] proposed a multi-scale network working in combination with a discriminator module. The network has been trained combining three losses, one regarding the reconstruction performance, one enforcing a smoothness constraint, and an adversarial one. The architecture has then been tested on the synthetic dataset they introduced. Another dataset was introduced by Guo et al [45], together with a deep learning model able to tackle both MPI and shot noise, and that is able to handle dynamic scenes too. Their model consists of an encoder-decoder architecture combined with a kernel prediction network used to tackle the shot noise. Agresti et al. [21] observed that the information regarding the structure of the scene is particularly important for MPI correction and, in order to have a simple network with about $150k$ parameters, they built it with two branches, one capturing the details and the other focusing on the high level geometry of the image. A similar idea was employed in [46] where a pyramid network observes the MPI structure at multiple resolutions, putting then the information together for the final prediction. In [39] the authors aimed

at filling the gap between prediction on synthetic and real data, using an unsupervised domain adaptation approach. They took the model from [21] and trained it as a GAN on unlabelled real data, clearly outperforming the original approach. The idea was later expanded by the same authors in [47], where they examined the possibility of performing domain adaptation also at input and feature level.

The use of the transient light model in Equation (3.3) as a prior for MPI correction is quite new in the literature. Barragan et al. [1] worked on the Fourier domain, using a U-Net architecture that takes a two-frequency input and predicts MPI corrupted data at several frequencies. They then compute the inverse Fourier transform on the output data, perform some filtering and get the depth prediction using a peak finding algorithm. The method shows good shot noise and MPI denoising capabilities but is quite heavy, with around $1.8M$ parameters.

Works directly targeted at the task of transient reconstruction from iToF information are very few, all using strong simplifying assumption for their solutions. Heide et al. [48] used an iToF camera to recover the depth information of a scene using the light reflected by a diffuse surface. They treated transient recovery as an optimization problem, constrained their solution both regarding spatial gradients and height field and introduced an algorithm in order to solve it. Lin et al. [49], showed that the information recovered from a multi-frequency iToF camera corresponds to the Fourier transform of a transient image. They then proposed an algorithm for transient reconstruction from a high number of iToF modulation frequencies. On a different note, Liang et al. [50] devised a deep learning model for the compression of rendered transient data, an important task due to the high volume of the data and the large amount of rendering noise.

In the next chapter we will introduce the datasets employed for training and testing our approaches, focusing in particular on the transient dataset that we have built.

4 Datasets

In this chapter we present the main datasets employed for the training and evaluation of the methods that we will propose. We employ both depth and transient datasets, due to the close relation between the two topics and the lack of datasets of the second kind. In particular, we introduce the *Walls* dataset: a novel synthetic transient dataset based on simple structures which has been used for training and evaluating the *Transient Reconstruction Module*.

4.1 iToF Datasets

Regarding the iToF data, we will mainly focus on the synthetic and real datasets introduced in [21, 39]. These datasets come with amplitude and phase information at three different modulation frequencies: $20, 50$ and 60 MHz; the scenes depicted are simple indoor scenes, with a maximum distance smaller than 7.5 *m* (the ambiguity range of the 20 MHz component) and high amounts of MPI. In particular, the synthetic dataset S_1 is composed of 54 scenes (40 for training and 14 for testing), has a high degree of shot noise and a spatial resolution of 240×320, while S_3, S_4 and S_5 are all real datasets with 8 images each, a limited amount of shot noise and a spatial resolution of 239×320. Dataset S_1 will be employed for training, S_3 for validation and S_4 and S_5 will be the main test sets for benchmarking the MPI correction capabilities of our network. A few examples from each dataset can be found in Figure 4.1. The iToF2dToF dataset [1] will instead be used for some additional studies regarding the resilience to shot noise and MPI correction. The dataset is composed of a total of 5000 images with a spatial resolution of 120×160 and with all iToF measurements ranging from 20 to 600 MHz with a step of 20. The dataset presents an extremely high amount of shot noise (around 80% of the total noise) and will be used as a stress test for our architecture. The scenes

https://lttm.dei.unipd.it/paper_data/transientMPI/

Dataset	# of images	Image type	Noise type	Ground truth
S_1 [21]	54	Synthetic	Shot	Depth
S_3 [39]	8	Real	Real	Depth
S_4 [21]	8	Real	Real	Depth
S_5 [39]	8	Real	Real	Depth
iToF2dToF [1]	5000	Synthetic	Shot,read	Depth*
FLAT [45]	2000	Synthetic	Shot	Transient
Walls	222	Synthetic	None	Transient

Table 4.1: Comparison between the employed datasets.
(*) transient available only for a small amount of data. Examples are shown in Figures 4.1 and 4.2.

from the iToF2dToF dataset are complex indoor scenes, and a couple of examples can be seen on the far right column of Figure 4.1.

4.2 Transient Datasets

The task of transient reconstruction is quite new in the literature and this is also due to the difficulties in acquiring a reliable transient dataset for the task. No real transient datasets are available and only few synthetic ones are freely accessible such as the FLAT dataset [45] and the Zaragoza [51] one. The iToF2dToF dataset [1] has both depth and transient ground truth, but the latter has been released only for a few images.

4.2.1 The Walls Transient Dataset

We introduce the *Walls* dataset: a synthetic transient dataset based on simple geometries. The dataset has been simulated using the Microsoft ToF Tracer [52] with a maximum depth set to 5 m. The simulated scenes consist of one to three walls with varying angles between them. The dataset has been built as a template case for MPI. The scenes, while very simple, still capture some of the most common MPI scenarios where the overestimation is due to at maximum a couple of reflecting surfaces. This assumption may not be true in general, but it is a good approximation for most practical cases, as the light intensity is inversely proportional to the square of the travelled distance, making the contributions of longer paths mostly negligible. In total, the dataset is com-

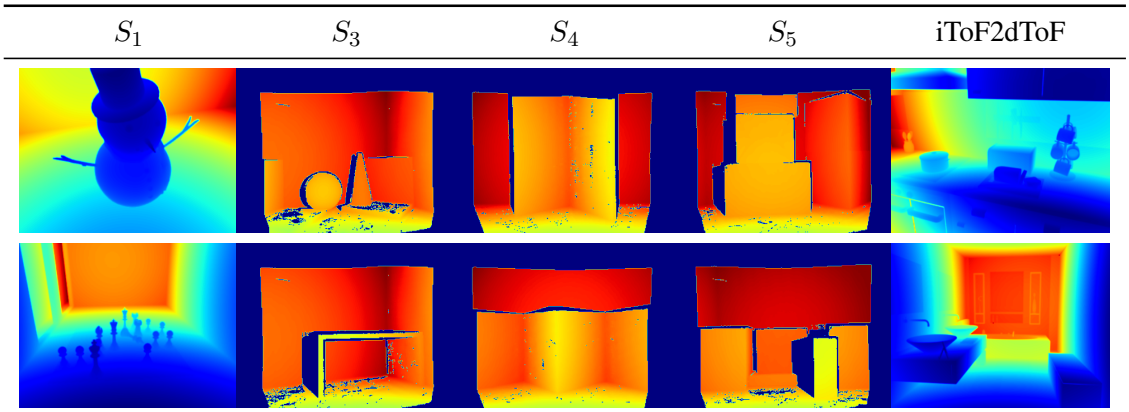
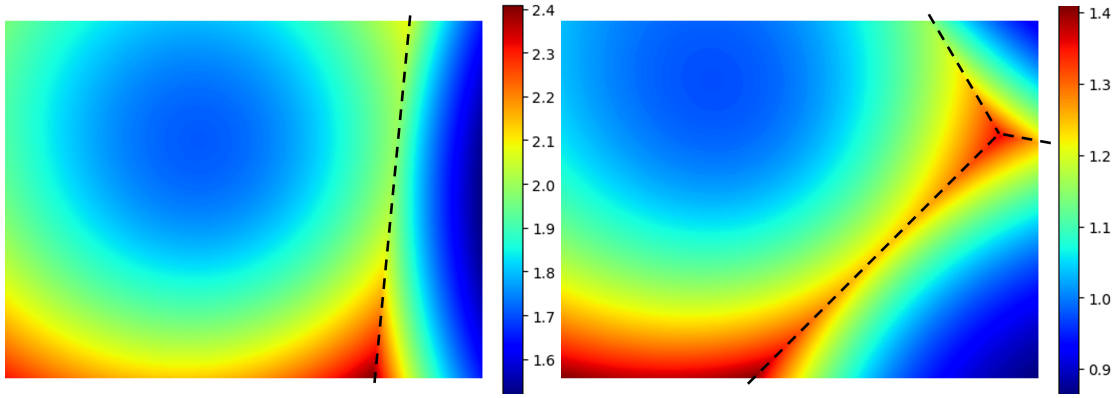


Figure 4.1: Sample images from the employed datasets.



(a) Two walls **(b)** Three walls

Figure 4.2: Depth images from our transient dataset.

posed of 222 images, 53 with a single wall, 95 with two, and 74 with three. A couple of samples can be seen in Figure 4.2.

The spatial resolution of our images has been set to of 480×640, to match that of some of the most recent ToF cameras, while the temporal dimension has been divided into 2000 bins; keeping into account that the maximum depth is 5 *m*, this means that the depth quantization step consists of 2.5 *mm*, a desirable property for indoor settings. The dataset has no noise sources other than MPI and rendering noise.

As the maximum depth of the *Walls* dataset amounts to 5 meters, while the one of S_1 gets to 7.5 meters, we decided to perform some data augmentation on the *Walls* dataset in order to cover the wider range. In practice, we added a shift to each of the transient vectors, randomly picking it from a uniform distribution in the $[0, 5]m$ range and from these we then recomputed the iToF measurements. The dataset can be found

at https://lttm.dei.unipd.it/paper_data/transientMPI/.

In the next chapter we will introduce a couple the methods for MPI correction and transient reconstruction and then provide a thorough evaluation against other approaches from the literature.

5 Methods for MPI Correction and Transient Reconstruction

In this chapter we present a few works on MPI correction and transient reconstruction. The common trend in the literature, that we introduced in Chapter 3, is to try and capture the high-level structure of the scene since it is strongly related to MPI. This is usually done either by using a wide receptive field for the network, or by employing multiple networks, each focusing on the scene at a different resolution level. The works that we introduce here instead, have the peculiarity of having a receptive field extremely small. The main source of information that we exploit is the change in MPI at different modulation frequencies. As we show in the results section, this information is enough to reach state of the art performance and allows at the same time a huge reduction in the number of network parameters, making our architectures extremely light.

As follows, we first give an initial introduction to the high-level structure, which is common to all approaches, and then we explain the *Two-Peaks Network*, our first approach for MPI correction. Afterwards, we show the architecture of the *Direct-Global Separation Network*, explain the improvements w.r.t. the first one and finally show some results for both methods for MPI correction and transient reconstruction. We then draw our conclusions for these two topics.

5.1 Network Structure

Deep neural networks can have issues when handling high-dimensional data [53, 54], and this is exactly the case of transient information; a backscattering vector can easily have a few thousand entries in the temporal direction against a handful of iToF measurements, as introduced in Chapter 3, a problem that can make the training of the architecture hard if not impossible. In order to solve this issue, we decided to split the backscattering estimation task in two parts as shown in Figure 5.1. On one side

we have the Backscattering model, which takes care of the dimensionality reduction, while on the other we can see the Predictive model, the true deep learning backbone of the approach. The learnable predictive model maps the iToF measurements into a low dimensional space that is then expanded into the transient information by the Backscattering model. This allows to greatly reduce the dimensionality of the deep network output space making the training of the model feasible. As follows, we will describe the two components.

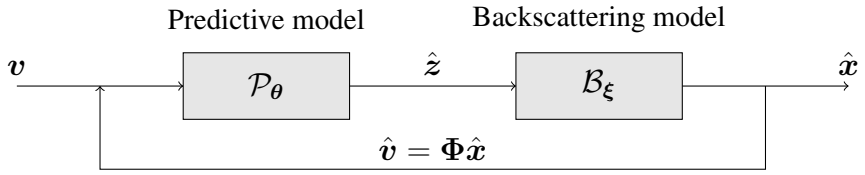


Figure 5.1: Structure of the proposed approach

5.1.1 Backscattering model

The main task of the *backscattering model* is to compact the high dimensional transient information into a representation that is easier to handle. Basically, the task of this module $\mathcal{B}_{\boldsymbol{\xi}}$ is to map a latent variable $\mathbf{z}$ into the respective backscattering vector $\boldsymbol{x}$:

$$\mathcal{B}_{\boldsymbol{\xi}}: \quad \mathbb{R}^L \rightarrow \mathcal{D}_{\boldsymbol{x}} \subseteq \mathbb{R}^N, \tag{5.1}$$

$$\boldsymbol{z} \rightarrow \boldsymbol{x} = \mathcal{B}_{\boldsymbol{\xi}}(\boldsymbol{z}), \tag{5.2}$$

where $L \ll N$, $\boldsymbol{\xi}$ are in principle some trainable parameters and $\mathcal{D}_{\boldsymbol{x}}$ is the domain of all possible backscattering vectors. In more general settings, $\mathcal{B}_{\boldsymbol{\xi}}$ could be for example a parametric model or an Autoencoder offering a precise mapping between a low-dimensional domain and the transient data.

5.1.2 Predictive model

The *predictive model* takes in input the matrix of raw iToF measurements at different modulation frequencies and outputs the corresponding values in the latent domain $\mathcal{Z}$. In practice, the predictive model is a highly non-linear function $\mathcal{P}_{\boldsymbol{\theta}}(\cdot)$, with parameters

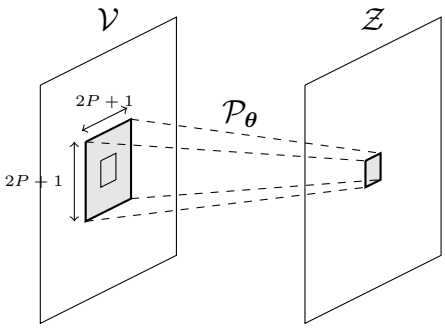


Figure 5.2: Predictive model working at local level

$\boldsymbol{\theta}$, that takes in input the vector $\boldsymbol{v}$ and produces an estimation of the corresponding vector $\boldsymbol{z}$, that we will call $\hat{\boldsymbol{z}}$.
In order to better exploit the spatial information for the prediction on each pixel, we consider a local neighbourhood around the pixel itself of size $(2P+1) \times (2P+1)$ as in Figure 5.2, with P a small value (i.e. for most of our trainings P is set to either 1 or 5, giving us a receptive field of respectively 3×3 and 11×11) .

5.2 Two-Peaks Network

The first method that we introduce is the *Two-Peaks Network*, the first model in the literature employing transient information inside the training pipeline. We will now describe the two components of the approach: first the Backscattering model, which in this case is a simple yet effective deterministic mapping, then the Predictive model, the deep learning backbone.

5.2.1 Backscattering model

For this method implementation, we decided to use a simple model for the backscattering vector, where just the two-rays interfering case is taken in consideration. This choice is motivated by the practical consideration that in real scenarios the first and second order reflections are the ones containing the main part of the energy of the backscattering vector [56]. For this reason, we used as backscattering model a deterministic mapping from a 4-dimensional $\boldsymbol{z}$ vector ($\boldsymbol{z} \in \mathbb{R}^4$) to a compressed version of the backscattering information. More in detail, the 4 values of the $\boldsymbol{z}$ representation are the amplitudes and the path lengths of the first and the second interfering rays. The

backscattering model has the task of converting these 4 values to the approximated backscattering vector that will be equal to zero on each entry apart from two peaks related to the first and the second interfering rays.

5.2.2 Predictive model

In Figure 5.3, we show the shape of the proposed network. We employ a Convolutional Neural Network whose first layer combines a weight kernel which retrieves information from each pixel, and another small $(2P+1) \times (2P+1)$ kernel centred around the pixel itself which takes care of the local information. The rationale behind this model is that even though local spatial information is quite important, at the same time it is crucial to give a great importance to the data carried by the central pixel itself. As we will show in Section 5.4, the spatial information ensures a slightly better performance, but the central pixel itself conveys already a degree of information which is sufficient for an accurate prediction. We want to bring to the attention of the reader that the strength of

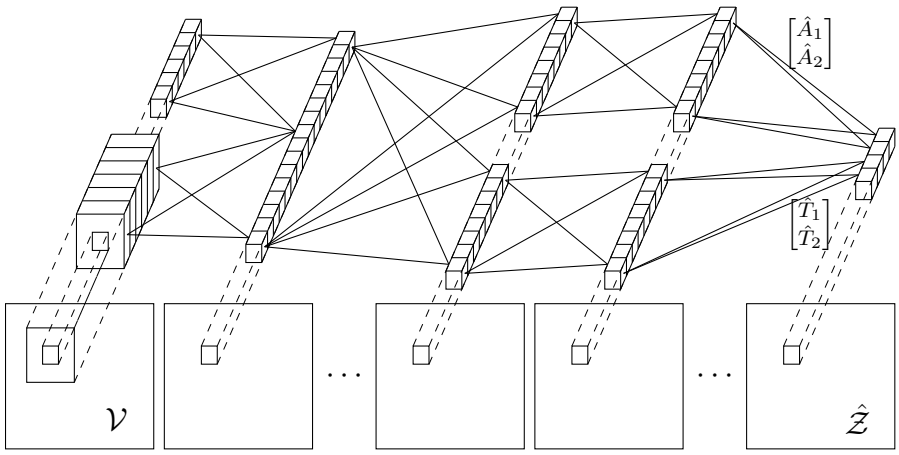


Figure 5.3: Predictive model structure

the approach does not rely for the most part on the quite reduced local information we provide (the kernels we employed are only 3×3) but on the global information that is inherently carried by each pixel. Other works in the literature relied on quite complex training structures in order to consider this critical piece of information as for example in [21], where a coarse network was proposed that considers the global structure of the scene, or in [46], where a pyramid structure tries to predict MPI at different resolution levels. The approach we propose does not need any complex addition, since the

global information is already present in the form of transient data and already conveys the information regarding the scene structure. Even if the two-peaks representation we consider may seem quite rough, it is still enough for an accurate depth estimation as we will prove in Section 5.4.

5.2.3 Training the Architecture

The training has been performed using a combination of two losses: a standard supervised loss and a soft constraint which made sure the predictions were consistent with the model defined in Equation (3.3). The latter, called measurement loss, ensures that our prediction makes sense according to the raw iToF measurement we gave in input. Since the matrix $\boldsymbol{\Phi}$ is known, for any predicted backscattering vector $\hat{\boldsymbol{x}}$, we can compute the corresponding vector $\hat{\boldsymbol{v}}$, which must be equal to the one we had in input for the prediction to make any sense. In other words we can write:

$$\mathcal{L}_m(\boldsymbol{v}, \boldsymbol{\Phi}\hat{\boldsymbol{x}}) = \|\boldsymbol{v} - \boldsymbol{\Phi}\hat{\boldsymbol{x}}\| = \|\boldsymbol{v} - \hat{\boldsymbol{v}}\|. \tag{5.3}$$

This loss of course only ensures a soft constraint since as mentioned the problem of Equation (3.3) is quite ill-conditioned. In order to get a meaningful result we therefore make use of a reconstruction loss, which simply ensures that our prediction $\hat{\boldsymbol{x}}$ matches with the ground truth $\boldsymbol{x}$. While being a simple supervised loss, it still presents a non trivial challenge as it is not that straightforward to define a suitable distance measure between sparse, high-dimensional vectors. Some common choices like the MSE or MAE quickly fail as we only have two meaningful values along a plateau of null entries. We need to define a loss function which amplifies the error in the case predicted and true peaks have different time and intensity components, and that at the same time keeps in lower consideration the null entries.
The main idea is to treat the two backscattering vectors as two different Probability Mass Functions (PMFs) and measure their statistical distance. The two distributions are defined as follows:

$$p_{\boldsymbol{x}}(n) = \frac{x_n}{X_{sum}} \qquad p_{\hat{\boldsymbol{x}}}(n) = \frac{\hat{x}_n}{X_{sum}} \qquad \text{where} \qquad X_{sum} = \sum_{n=0}^{N-1} x_n, \tag{5.4}$$

where we normalize for the ground truth values in order to avoid divisions by zero issues when we have all-zero predicted vectors. The distance between the two is then

computed using a modified version of the Earth Mover Distance (EMD) which, unlike some other divergence measures such as the Kullback-Leibler or Jensen-Shannon ones, does not require the two PMFs to have a common support. The standard EMD is defined as follows:

$$EMD(p_{\boldsymbol{x}}, p_{\hat{\boldsymbol{x}}}) = \sum_{n=0}^{N-1} |P_{\boldsymbol{x}}(n) - P_{\hat{\boldsymbol{x}}}(n)|, \tag{5.5}$$

where $P_{\boldsymbol{x}}(n)$ and $P_{\hat{\boldsymbol{x}}}(n)$ are the cumulative mass functions of the original distributions. Starting from the previous expression, we define the reconstruction loss between original and predicted backscattering vector according to a weighted Earth Mover Distance (EMD_w) as below:

$$\mathcal{L}_r(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{NX_{sum}} EMD_w(p_{\boldsymbol{x}}, p_{\hat{\boldsymbol{x}}}) = \frac{1}{NX_{sum}} \sum_{n=0}^{N-1} w_n |c_n - \hat{c}_n|, \tag{5.6}$$

with c_n and $\hat{c}_n$ the cumulative functions of our backscattering vectors:

$$c_n = \sum_{n'=0}^{n} x_{n'} \qquad \hat{c}_n = \sum_{n'=0}^{n} \hat{x}_{n'}, \tag{5.7}$$

while the weights w_n are computed as:

$$w_n = \frac{1}{W} \sum_{k=0}^{W-1} |c_{n-k} - \hat{c}_{n-k}|, \tag{5.8}$$

with W a suitably sized window that in our experiments was set to 100. The reason for this modification is due to the fact that it is quite hard for the network to distinguish between direct and global component when the two peaks are very close one to the other; what tends to happen is that the first peak obscures the other, leading to predicting a single peak. The solution proposed in Equation (5.8) consists in giving more importance to elements which are preceded by other non-zero samples, thus balancing out the importance given to direct and global component.

5.2.4 Bilateral Filtering

To further improve performances, an additional step is included in the pipeline in order to deal with zero-mean error sources. In practice, the predicted depth goes through

a bilateral filter thus giving us the final prediction. The parameters of the filter were experimentally set to $\sigma_d = 0.05$ and $\sigma_s = 10$, where the first value corresponds to the kernel in the depth domain, and the second to the spatial one instead.

5.2.5 Training and Test Datasets

For the supervised optimization of the proposed approach we need a training set containing raw ToF data together with the corresponding ground truth transient data. Note that from geometrical considerations it is clear that the true depth value is always associated to the shortest returned path that corresponds to the direct component, and therefore depth information can be easily extracted from the transient scene. The acquisition of a real dataset with transient ground truth however is a quite complex and time consuming task; since no publicly available datasets of the kind exist, we had to rely on synthetic data.

For the training of the approach we relied on the FLAT synthetic dataset introduced in [45], which contains transient data. At first, we applied a depth equalization procedure, in order to obtain a final distribution that is as uniform as possible. Then, the data was processed as discussed in Section 5.1.1 with the addition of a clipping operation for the intensity of the second peak, whose maximum value could be $h_2 \leq 0.8h_1$, with h_1 and h_2 the intensity of the two peaks. Finally, the input iToF values were computed from the compressed transient information using the measurement model described in Equation (3.3) with modulation frequencies of $20, 50$ and 60 MHz. After the processing the data was then split into a training and a validation set, made of 211200 and 2064 3×3 patches respectively. Note that no test set was built at this phase since the testing will be performed on real images.

The final performance in the depth estimation provided by the proposed approach is evaluated on real-world scenes where no transient data is given. Since our objective is MPI denoising, transient data is not required at the testing phase, all that is needed are the raw measurements at the desired modulation frequencies and the corresponding ground truth depth maps. From the predictions of our dataset, we can focus on the first peak and use that to estimate the depth value of each pixel. The real-world datasets on which we carry out our analysis are the three real ToF datasets S_3, S_4, S_5 provided by Agresti et al. in the works [21, 39]. All three datasets have been captured in a laboratory environment without external illumination using the SoftKinetic ToF camera DS541

at multiple modulation frequencies. For each scene they provide unwrapped phase, amplitude and intensity, as well as depth ground truth. The datasets are in the depth range between 58 and 203 cm.
In Table 5.1 we can see the resolution, number of and acquired modulation frequencies of the three datasets. In particular, dataset S_3 will be used for validation while S_4 and

Dataset	Type	Depth GT	Trans. GT	No. Scenes	Spatial Res.	Modulation frequencies
S_3	Real	yes	no	8	320×239	10,20,30,40,50 and 60 MHz
S_4	Real	yes	no	8	320×239	20,50 and 60 MHz
S_5 (*box*)	Real	yes	no	8	320×239	10,20,30,40,50 and 60 MHz

Table 5.1: Properties of the real-world datasets S_3, S_4 and S_5.

S_5 will be our test sets.

5.3 Direct-Global Subdivision Network

The second method is made of a light and modular architecture. It exploits the subdivision between direct and global components of transient vectors to reach state-of-the-art performance both for MPI removal and transient reconstruction. We start by describing the idea behind it and then go in detail through each of the three components of the architecture: the *Spatial Feature Extractor*, the *Direct Phasor Estimator* and the *Transient Reconstruction Module*. Referring to Figure 5.1, the first two models correspond to the Predictive model, while the Backscattering model is the *Transient Reconstruction Model*. In the end of the section, we introduce the losses employed for training.

5.3.1 Direct-Global Subdivision

Let's begin by considering the structure of a common transient vector (see the example in Figure 3.1); it is quite clear that it is composed of two quite distinct parts: one corresponding to the first peak, the other instead incorporating all the other incoming light rays. From now on we will denote the *direct component* composed by the first peak alone, as the vector $\boldsymbol{x_d}$, while the *global component*, made of all the other reflections, will be denoted by $\boldsymbol{x_g}$. We can now consider Equation (3.3) and write

$$\boldsymbol{v} = \boldsymbol{\Phi x} = \boldsymbol{\Phi}(\boldsymbol{x_d} + \boldsymbol{x_g}) = \boldsymbol{v_d} + \boldsymbol{v_g}, \tag{5.9}$$

where we exploited the linearity of the model to extract the v_d and v_g vectors. What follows from this derivation is that the subdivision of the transient vector into direct and global components can be translated also onto the iToF domain. In practice, we now have a vector v_d which corresponds to ideal iToF measurements, the ones that would be produced by the direct peak alone, while v_g are the measurements corresponding to all reflections but the first.

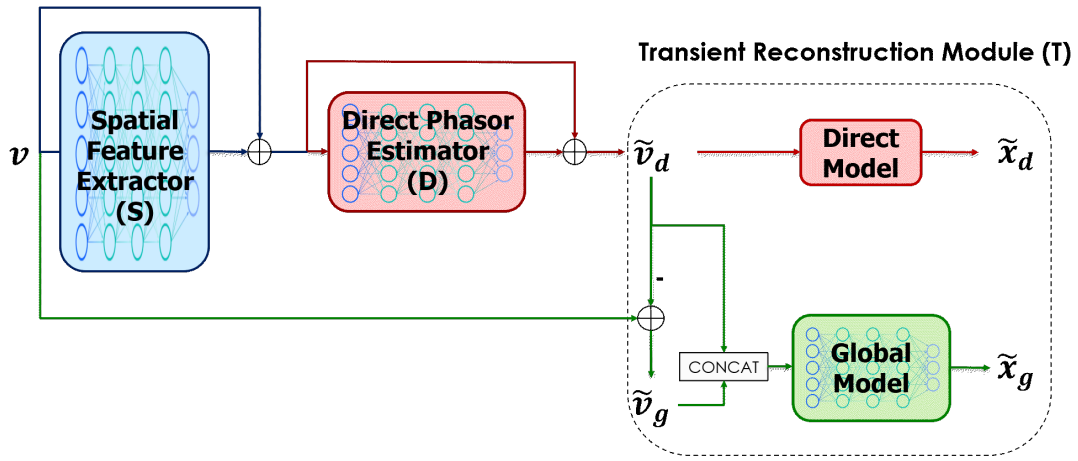


Figure 5.4: High level structure of our training architecture

5.3.2 Deep Learning Architecture

The different modules of our network are shown in Figure 5.4. As input the model takes in the real and imaginary components of the raw iToF measurements v at different modulation frequencies. First, they go through the *Spatial Feature Extractor*, which exploits the spatial information to produce an intermediate representation of the data (it proved to be very useful for handling zero-mean noise). Its output is then processed by the *Direct Phasor Estimator*, that predicts the iToF measurements corresponding to the direct component, which, subtracted from the original input, gives us also the iToF measurements corresponding to the global component. The two predictions are in the end fed to the *Transient Reconstruction Module* that has the task of reconstructing the whole transient vector. As we will see, this module is further split into the *Direct Model* which is a deterministic function computing the direct component, and the *Global Model* that instead consists of a deep learning architecture predicting the global

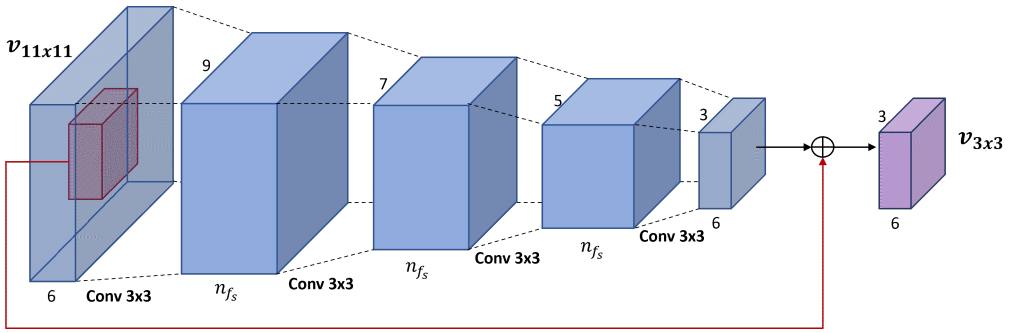


Figure 5.5: Representation of the Spatial Feature Extractor module for 3 input frequencies and an input patch size of 11×11. The number of feature maps n_{f_s} is equal to 32 for all experiments.

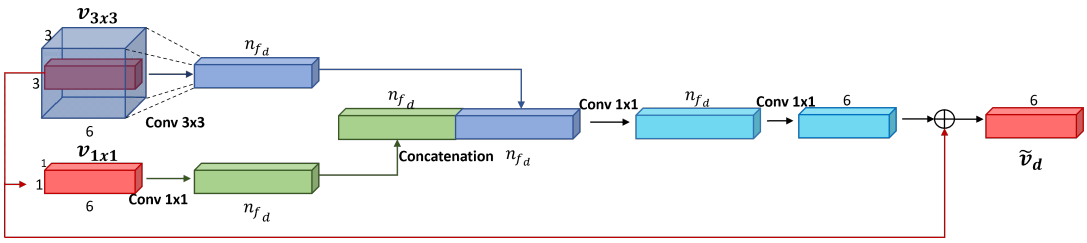


Figure 5.6: Representation of the Direct Phasor Estimator module (upper part) for 3 input frequencies. The number of feature maps n_{f_d} is equal to 8 when both the S and D models are used, and to 32 when the D model is employed alone.

component. For the construction of the learning model we used as a starting point the network introduced in [22], where the raw iToF input was directly mapped into an oversimplified encoded version of a transient vector, consisting of two peaks. We kept a narrow receptive field claiming that the information in the transient dimension is enough for MPI correction, but differently from [22], we introduce an intermediate training target between the input $\mathbf{v}$ and the transient prediction $\tilde{\mathbf{x}}$ (i.e., the subdivision into direct and global components). Moreover, we introduce a more complex and realistic model for the backscattering vector itself.

Spatial Feature Extractor (S)

The main task of this module is providing an encoded version of spatial information to the following stages. As we will see, the *Direct Phasor Estimator* has a very narrow receptive field (i.e. 3×3), which limits its capability of managing noise sources such as shot noise. The *Spatial Feature Extractor* is a fully convolutional architecture with

a 9×9 receptive field. It consists of 4 layers, each with $n_{f_s} = 32$ feature maps and a residual connection links the central 3×3 part of the input to the output. A visual representation of this network can be found in Figure 5.5.

DIRECT PHASOR ESTIMATOR (D)

This module estimates $\boldsymbol{v_d}$, the direct component of the raw phasor. The raw measurements coming from the *Spatial Feature Extractor* are fed to two branches with receptive field 3×3 and 1×1 respectively, whose outputs are then concatenated and used for the prediction of $\tilde{\boldsymbol{v}}_{\boldsymbol{d}}$. More in detail, as depicted in Figure 5.6, it takes in input both a 3×3 patch and its central pixel; they go through a convolutional layer with an output of size 1×1 and are then concatenated. The information is then processed by two other convolutional layers before producing the $\tilde{\boldsymbol{v}}_{\boldsymbol{d}}$ prediction. Each convolutional layer has n_{f_d} feature maps and there is a residual connection between input and output. From the prediction of $\tilde{\boldsymbol{v}}_{\boldsymbol{d}}$ we then compute the corresponding depth for each of the modulation frequencies, using the smallest frequency for solving ambiguity range uncertainty on the higher ones. The output depth maps are then passed through a bilateral filter and the final depth prediction will be the pixel-wise minimum of the output depths. The reason for this is that the MPI, that is the major cause of error, leads to an overestimation of the distance. Considering Equation (5.9) we can then retrieve also the iToF measurements corresponding to the global component by simply subtracting the direct component, i.e., $\tilde{\boldsymbol{v}}_{\boldsymbol{g}} = \boldsymbol{v} - \tilde{\boldsymbol{v}}_{\boldsymbol{d}}$. Notice that this module is tackling the MPI removal task, as if the direct-global subdivision is successful, we are able to recover an MPI-free estimate of our input from the $\tilde{\boldsymbol{v}}_{\boldsymbol{d}}$ component. The number of feature maps is set to $n_{f_d} = 8$ when the S and D models are used together and to $n_{f_d} = 32$ when the D model is used alone.

TRANSIENT RECONSTRUCTION MODULE (T)

Retrieving the transient information from iToF leads to some serious challenges, not only linked to the difficulty of the task itself, but also to the dimensionality of our output. As remarked in Chapter 3, we want to map the raw iToF measurements, that correspond to a handful of values, into a vector with thousands of entries. The complexity of the matter makes an encoding of the ground truth a necessity. Since the one proposed in [22] is way too simplistic, and the one by Liang et al. [57] computationally

heavy, we propose a novel approximation of the transient vector $\boldsymbol{x_g}$ with just 6 parameters, 2 needed for the direct component, and the other 4 for the global. Therefore, the *Transient Reconstruction Module* is further split into two components, the *Direct Model* that takes care of the reconstruction of the direct component and the *Global Model*, which instead predicts the global component.

DIRECT MODEL Similarly to [22], each direct component $\boldsymbol{x_d}$ gets encoded by its magnitude E_d and time position t_d. As a matter of fact, no learnable parameters are needed for the prediction of the direct component $\boldsymbol{x_d}$, since the time position t_d is directly proportional to the phase φ_d through Equation (2.9), and can be retrieved directly from $\tilde{\boldsymbol{v}}_{\boldsymbol{d}}$. At the same time, the magnitude of the first peak E_d is strictly related to the amplitude of the raw iToF measurements of the direct component; this is true since in the case of a single peak, the magnitude is the value of the peak itself, while the amplitude A_d can be written as follows,

$$\begin{aligned} A_d &= \frac{1}{2}\sqrt{\boldsymbol{v_{d,\Re}}^2 + \boldsymbol{v_{d,\Im}}^2} \\ &= \frac{1}{2}\sqrt{\left(\sum_{t=0}^{T} \Phi_{\Re,t}\boldsymbol{x_t}\right)^2 + \left(\sum_{t=0}^{T} \Phi_{\Im,t}\boldsymbol{x_t}\right)^2} = \\ &= \frac{1}{2}\sqrt{\left(\Phi_{\Re,t_d}x_{t_d}\right)^2 + \left(\Phi_{\Im,t_d}x_{t_d}\right)^2} = \\ &= \frac{1}{2}x_{t_d} = \frac{1}{2}E_d, \end{aligned} \tag{5.10}$$

where we used the Pythagorean identity and with the fact that only one element of the sum is non-zero (the one at time index t_d). $\boldsymbol{v_{d,\Re}}$ and $\boldsymbol{v_{d,\Im}}$ are the real and imaginary components following the phasor notation in Equation (2.10).

GLOBAL MODEL For the encoding of $\boldsymbol{x_g}$ we chose instead the following parametric function $\tilde{\boldsymbol{x}}_{\boldsymbol{g}}(t)$ inspired by the Weibull distribution [58]

$$\tilde{\boldsymbol{x}}_{\boldsymbol{g}}(t) = \mathbf{a}(t-\mathbf{b})^{\mathbf{k}-1}\exp\left(-\frac{t-\mathbf{b}}{\boldsymbol{\lambda}}\right)^{\mathbf{k}}, \tag{5.11}$$

where t ranges from 0 to T (the maximum acceptable travel time), $\mathbf{a}$ takes care of the scale, $\mathbf{b}$ of the shift, and $\mathbf{k}$ and $\boldsymbol{\lambda}$ of the shape. For the choice of this function we took

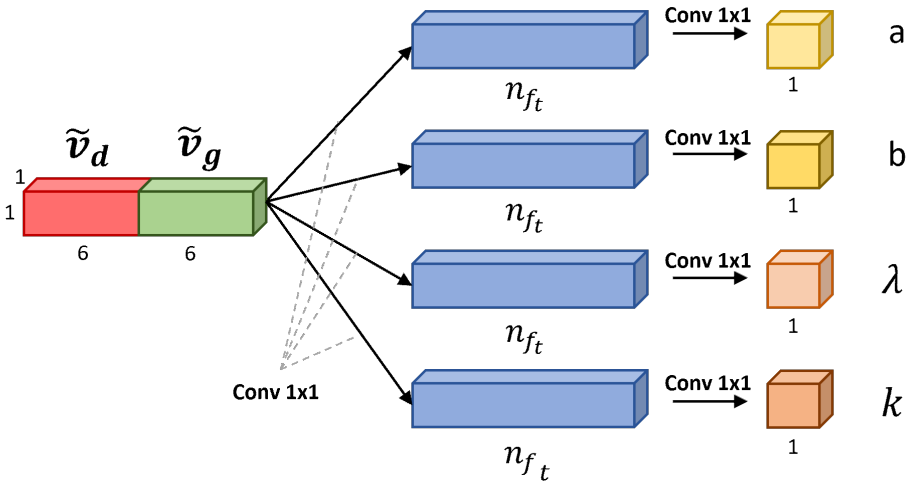


Figure 5.7: Representation of the *Global Model*. The number of feature maps n_{f_t} has been set to 32 for all experiments.

inspiration from the topic of multipath interference related to radio signals, where distributions such as the Rayleigh or the Weibull are usually employed [58]. In the end we decided to employ the Weibull distribution since it is a generalization of the Rayleigh and shows a good resemblance with common shapes of transient vectors. Predicting the parameters of the global component $\tilde{\boldsymbol{x}}_g$ expressed in Equation (5.11) from $\tilde{\boldsymbol{v}}_g$ and $\tilde{\boldsymbol{v}}_d$ is a quite complex task, which is handled by an additional deep learning architecture. The *Global Model* is composed of 4 parallel branches with a 1×1 receptive field, each predicting one of the 4 parameters of the parametric function. Each branch is composed of a stack of 2 convolutional layers with a total of 32 feature maps. It takes $\tilde{\boldsymbol{v}}_g$ and $\tilde{\boldsymbol{v}}_d$ in input, estimates from it the 4 parameters of the function described in Equation (5.11), and finally compares it to the ground truth $\boldsymbol{x}_g$. The *Global Model* can be seen in Figure 5.7. The proposed model for global prediction is a clear improvement w.r.t. the competitors as the Weibull function provides a much better fitting of a distribution such as the one in Figure 3.1 than the single peak prediction of [22], which compacts all the information in a single bin. Combining together the outputs of the *Direct Model* and of the *Global Model*, we obtain an estimate of the transient vector. Notice that while the proposed reconstruction of the global component of light is more advanced than the ones of previous works, still it only estimates a single Weibull function and therefore assumes any secondary reflection to come from a single surface. While this is a quite coarse simplification, it can still provide a sufficiently good recon-

struction for simple tasks such as tracking NLOS objects or material estimation.

5.3.3 Training Targets

The losses used for training our architecture are the Mean Absolute Error (MAE), and the Earth Mover's Distance (EMD) [59]. The *Direct Phasor Estimator* uses as guidance a simple MAE on the target values, while the EMD guides the training of the *Global Model*. The ground truth $\boldsymbol{v_d}$ can only be retrieved from the transient information and for this reason it is not available when using common iToF datasets. For this reason, we employed two different training methodologies according to the input data:

1. When the transient data is available we can directly compute the loss between the ground truth $\boldsymbol{v_d}$ and our prediction $\tilde{\boldsymbol{v}}_{\boldsymbol{d}}$ as

$$\mathcal{L}_{MAE_{v_d}} = \mathbb{E}\left[\left\|\boldsymbol{v_d} - \tilde{\boldsymbol{v}}_{\boldsymbol{d}}\right\|_1\right]. \tag{5.12}$$

2. When instead the dataset only offers the depth ground truth, we are unable to recover $\boldsymbol{v_d}$, but we are able to compute the ground truth phase delay $\boldsymbol{\varphi_d}$ following Equation (2.9). From the network prediction $\tilde{\boldsymbol{v}}_{\boldsymbol{d}}$ we can thus compute the predicted $\tilde{\boldsymbol{\varphi}}_{\boldsymbol{d}}$ through an arctangent operation, and finally compute the loss as:

$$\mathcal{L}_{MAE_{\varphi_d}} = \mathbb{E}\left[\left\|\boldsymbol{\varphi_d} - \tilde{\boldsymbol{\varphi}}_{\boldsymbol{d}}\right\|_1\right]. \tag{5.13}$$

Furthermore, if the training dataset instead contains not only the depth ground truth, but also images both with and without zero-mean noise, it is possible to extend the interpretability of the architecture, by pinning the output of the *Spatial Feature Extractor* with an additional loss. The *Spatial Feature Extractor* would therefore be dealing only with zero-mean noise, while the *Direct Phasor Estimator* would take care exclusively of MPI. The output of the *Global model* is guided instead by the EMD, which we had already employed in [22], and is defined as

$$\mathcal{L}_{EMD} = \mathbb{E}\left[\left\|\boldsymbol{X_g} - \tilde{\boldsymbol{X}}_{\boldsymbol{g}}\right\|_1\right], \tag{5.14}$$

where $\boldsymbol{X_g}$ and $\tilde{\boldsymbol{X}}_{\boldsymbol{g}}$ are the cumulative sums of $\boldsymbol{x_g}$ and $\tilde{\boldsymbol{x}}_{\boldsymbol{g}}$ respectively. What this distance measure captures is the dissimilarity between the two distributions, i.e., the minimum amount of work needed to convert one into the other [59].

The performance of the different losses and the modularity of the approach will be thoroughly investigated in Section 5.4.

5.4 Results

In this Section we are going to present some experimental results regarding the two previously introduced approaches. We will compare the two against the previous state-of-the-art and against each other, and finally show some ablation studies specific to each approach.

5.4.1 Training Details

The *Two-Peaks Network* was trained on the FLAT dataset processed as described in Section 5.2.5 using the Adam optimization algorithm. The entire dataset was divided into batches of 1024 samples each, and the gradient at each iteration was computed on a single batch. We run the training for a total number of $E = 2000$ epochs on a Nvidia GeForce GTX 1060 GPU. The overall time required was around 6 hours. To account for the noise always present in any real-world ToF measurements, at each iteration a gaussian zero-mean random noise is added to the real and imaginary parts of the simulated raw ToF data:

$$\boldsymbol{v} = \boldsymbol{\Phi}\,\boldsymbol{x} + \boldsymbol{\eta} \qquad\qquad \boldsymbol{\eta} \sim \mathcal{N}(0, \sigma_v^2) \tag{5.15}$$

The noise is independent and identically distributed across all the pixels in the image and across all the acquired phasors at different modulation frequencies. Changing the noise at each iteration helps avoiding overfitting and acts as a form of regularization. The network never sees the same exact input data more than once. Moreover, it helps the network in learning to denoise the input data, giving more importance to the more stable relationships between the acquired phasors and less to small fluctuations around the average. In Section 5.4.4 we will show this more in detail; for all other experiments, unless otherwise stated, we will use noise with a standard deviation $\sigma_v = 0.02$. The choice of this value was performed as mentioned in Section 5.4.4.
The best set of weights are chosen according to the network performance on the real dataset S_3, which we employ as a realistic validation set. The testing is then carried out from a qualitative and quantitative point of view on the two real datasets S_4 and

S_5 provided by Agresti et al. [21, 39]. As anticipated in Section 5.2.5, the evaluations focus on the degree of MPI correction, while a few comparisons concerning the reconstruction of the transient component are reported in Section 5.4.4. The metric used to quantify the error in the depth domain is the Mean Absolute Error (MAE): the lower the better.

The *Direct-Global Separation Network* has been trained using the S_1 and *Walls* datasets. From the first one we took the original training set of 40 images, while from the *Walls* dataset we randomly picked 134 images. The validation set consists instead of the 8 images from the real dataset S_3. In particular, the training data has been cut in patches of size 11×11, randomly chosen inside the images, while the validation set has been kept at full resolution. The models have been developed in Tensorflow 2.1, the trainings of our architecture have been performed on an NVIDIA 2080 Ti GPU, with ADAM as optimizer with a learning rate of 10^{-4} and a batch size of 2048. We will focus our evaluation for MPI correction on two models: the first one comprised of the first two modules introduced in Section 5.3 which we will abbreviate with *SD*, and the second a lighter architecture without the *Spatial Feature Extractor*. In this case the number of feature maps of the *Direct Phasor Estimator* was changed from 8 to 32 to provide the network with additional learning parameters. We will abbreviate this second model with *D*. In all cases, each input patch has been normalized by the mean amplitude of its 20 MHz component to help generalizing on real data.

5.4.2 Results on MPI Correction

We will now compare the two approaches with some of the best performing MPI correction methods. The comparison will be made with SRA [19], an algorithmic approach, with DeepToF [38], one of the first deep learning approaches for MPI correction, and with the approach from Agresti et al. [21], together with the subsequent domain adaptation approaches *in*-DA, *feat*-DA and *out*-DA proposed in [47].

In Table 5.2 we show the overall comparison between the cited approaches and the two proposed architectures. The first two columns show the MAE on the two real datasets S_4 and S_5, while the last one shows the network complexity of each approach; SRA has no entry as it isn't deep learning based. Regarding our approaches, the parameters of the *Transient Reconstruction Network* were not included in the total amount as it is not needed for MPI correction. Moreover, note that while the *SD* model has been

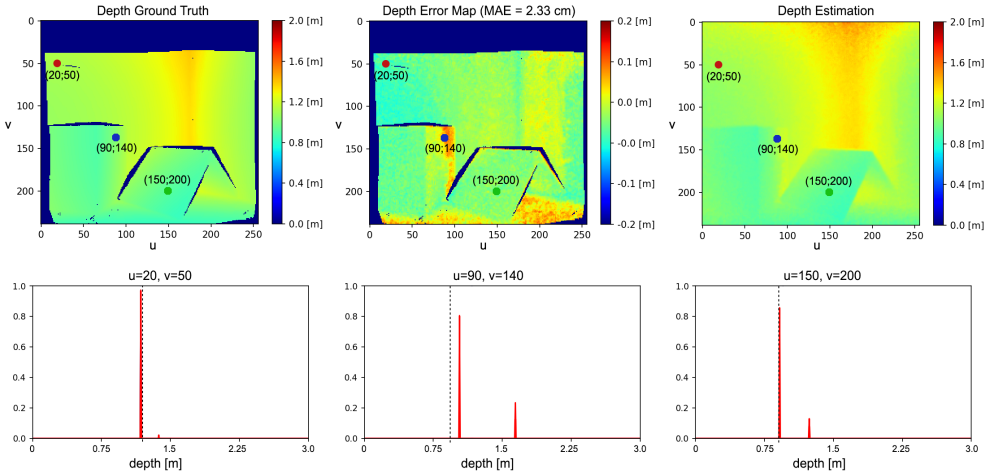


Figure 5.8: Network prediction for selected pixels in an image. The dashed lines correspond to the depth ground truth values while the red plots indicate the predicted backscattering vectors.

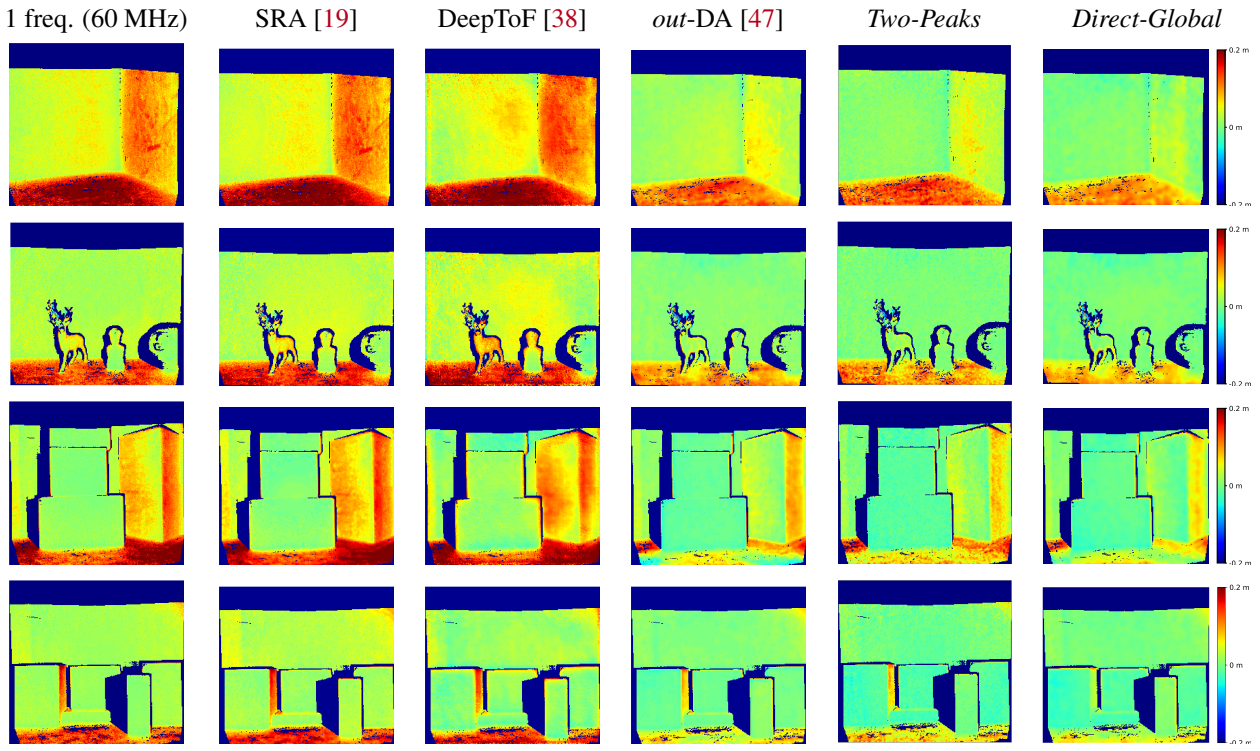


Figure 5.9: Qualitative comparison between some of the best approaches for MPI correction. The first two images come from the S_4 dataset, while the other two from S_5. All the images display the reconstruction error w.r.t. the ground truth, where green means good reconstruction and red an overestimation.

Approach	S_4		S_5		# of
	MAE $[cm]$	Relative error	MAE $[cm]$	Relative error	param.
Input (60 MHz)	5.43	-	3.62	-	-
Input (20 MHz)	7.28	-	5.06	-	-
SRA [55]	5.11	94.1%	3.37	93.1%	-
DeepToF [38]	5.13	70.5%*	6.68	132%*	330k
+ calibration	5.46	75%*	3.36	66.4%*	330k
Agresti et al. [21]	3.19	58.7%	2.22	**60.5%**	150k
+in-DA [47]	2.40	44.2%	1.74	48.1%	150k
+feat-DA [47]	2.37	43.6%	1.66	45.8%	150k
+output-DA [47]	2.31	42.5%	**1.64**	**45.3%**	150k
Two-Peaks Network [22]					
3×3 receptive field	2.79	51.4%	2.27	62.7%	22k
+ bilateral filtering	2.60	47.9%	2.12	58.6%	22k
1×1 receptive field	3.43	63.2%	2.52	69.6%	14k
+ bilateral filtering	2.99	55.0%	1.88	51.9%	14k
Direct-Global Network [23]					
D (*Walls*)	2.46	45.3%	1.98	54.7%	**3k**
D (*Walls*)	2.40	44.2%	1.88	51.9%	25k
SD (*Walls*+S_1)	**2.06**	**37.9%**	1.87	51.7%	23k

Table 5.2: Quantitative comparison between several state-of-the-art MPI correction algorithms on the real datasets S_4 and S_5. The evaluation metrics are the MAE and the relative error compared to the highest input frequency. * is compared to the 20 MHz input as it is single frequency. The best performing methods are highlighted in **bold**, while the best results for each of the two proposed architectures are underlined. The complexity of each method is also displayed.

trained on both the S_1 and *Walls* datasets, *D* has been trained on *Walls* alone, since as we will see it is not able to deal with shot noise due to its very narrow receptive field. The real datasets present a clear challenge due to the domain shift between synthetic and real data. In practice, the resemblance between training and test data is strictly limited by the accuracy of the simulation, which can mimic a real scenario only up to a certain extent.

On real-world scenes the *Two-Peaks Network* approach achieves performance comparable to the other state-of-the-art algorithms. It produces an error of 2.60 cm on S_4 and an error of 2.12 cm on S_5, performing better than most competing methods, with

Approach	S_1 $[cm]$	# of parameters
Single freq. (60 MHz)	16.7	-
SRA [55]	15.0	-
DeepToF [38] + calibration	26.1	330k
Agresti et al. [21]	7.49	150k
Two-Peaks Network [22]		
trained on FLAT dataset	30.5	22k
trained on the *Walls* dataset	20.0	22k
Direct-Global Network [23]		
D	12.2	**3k**
SD	**6.17**	23k

Table 5.3: Quantitative comparison between several state-of-the-art MPI correction algorithms on the test set of the synthetic dataset S_1. The evaluation metric is the MAE. The complexity of each method is also displayed.

the exception of the unsupervised domain adaptation technique of Agresti et al. [39] which produces test errors of respectively 2.36 and 1.66 cm. We stress the fact that the unsupervised domain adaptation technique has been adapted using unsupervised real data similar to the one in S_4 and S_5, while our approach relies only on a completely different synthetic training dataset. It is remarkable to notice that our method clearly outperforms the SRA method [55], which is the one adopting the most similar setup to ours. We both acquire data at three modulation frequencies and use a physical model to describe the MPI effect under the specular reflections assumption. The MAEs on datasets S_4 and S_5 obtained by SRA are respectively 5.11 and 3.37 cm. Even if in the case of *Two-Peaks Network* we are considering only two reflections, experimental results show good MPI compensation capabilities, confirming that many real-world cases can be well approximated by a two components reflection model. This is due to the fact that, since the light power decays with the square of the distance, higher order reflections reaching the camera are very dim. Moreover, real lambertian surfaces present always a fraction of specular reflections and thus our assumption in first approximation holds also for those surfaces. From Figure 5.9 it is possible to see how this approach removes most of the MPI on wall surfaces and reduces it in proximity of edges. The large amount of MPI on the floor surfaces is also consistently reduced even if some depth reconstruction errors are still present. These areas are probably subject to more

complex reflection patterns and are therefore more error prone. Looking at the network output at some significant points (Figure 5.8), it is evident that it has learnt to discriminate between MPI-free and MPI-affected pixels, introducting the global component only when it is necessary to compensate for the MPI effect and providing a more reliable depth estimation.
The *Direct-Global Separation Network* instead, not only clearly outperforms both the architecture proposed in [21] and the *Two-Peaks Network*, but also beats the results from [47] on S_4, while falling shortly behind on S_5, all by using just $\frac{1}{7}$ of the parameters from [47]. This is particularly striking as differently from [47] we only rely on synthetic data for our prediction. Given this, we have clear reasons to expect an even better performance by using some unsupervised domain adaptation techniques as was done in [47]. Another interesting outcome is the fact that the *D* module alone, shows quite competitive results w.r.t. *SD* and [47] and at the same time outperforms other architectures such as [21] and the *Two-Peaks Network*. The *D* model is extremely light, with just around 3k learnable parameters, but still gets close to state-of-the-art results. A second version of the D model with 25k parameters has also been trained but the improvement is not significant w.r.t. the lighter version.

Figure 5.9 shows a qualitative comparison on a few images from the S_4 and S_5 datasets. The *Two-Peaks Network* clearly outperforms the DeepToF approach, showing MPI correction capabilities close to those of [47] on difficult surfaces such as the floors or tilted surfaces. It is also possible to notice that the remaining artefacts show a salt and pepper kind of behaviour, which is due to the very small receptive field of the network. The limitations of this small receptive field can be seen also from Table 5.2, where the introduction of bilateral filtering significantly improves the performance of the network. The *Direct-Global Separation Network* we propose shows a clear improvement on the competitors, providing a good reconstruction also on regions highly corrupted by MPI such as the floor and other steeply sloped scene elements. As an example we can consider the first row of Figure 5.9, where not only the floor shows a better reconstruction, but the MPI artefacts on the wall on the right are almost completely corrected. Similar considerations can be made on the last image row, where our approach is the only one able to clear the right face of the box on the left side, a particularly difficult surface due to its tilt.

In Table 5.3 we report instead the comparison made on the S_1 dataset. This is interesting due to the abundant presence of shot noise which can hinder the performance

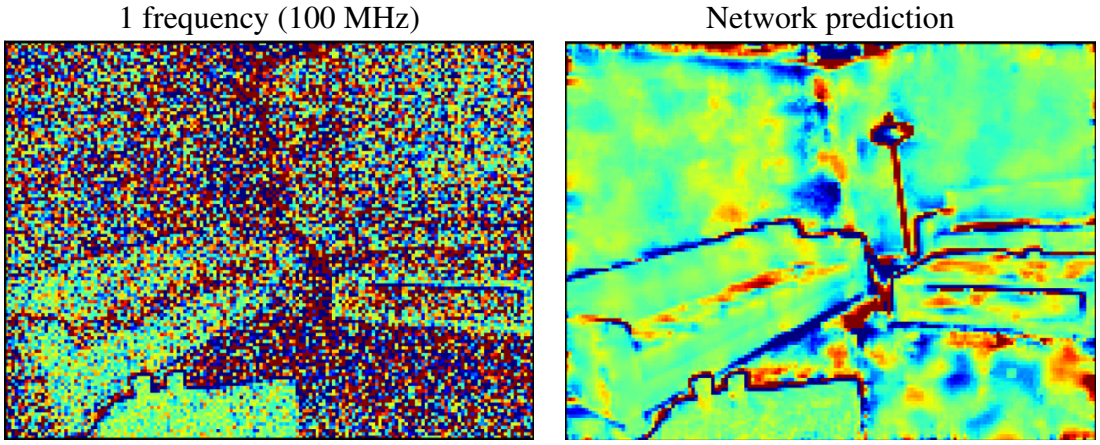


Figure 5.10: Qualitative results on a particularly noisy image from the test set of the iToF2dToF dataset [1]. On the left side the single frequency reconstruction at 100 MHz, on the right side the network prediction.

of some approaches. The problem is that, while MPI correction can be performed using only information along the transient dimension, that is not possible for shot noise removal. Networks such as the *Two-Peaks Network* and the *D* module of the *Direct-Global Separation Network* have a receptive field of size 3×3, which hampers the performance on S_1. We tested the *Two-Peaks Network* once training it on the FLAT dataset [45], and then on the *Walls* dataset (training it on S_1 is not possible due to the lack of transient information). In both cases, as expected, the performance is not satisfactory. Similar conclusions can be drawn when training the *D* model alone. Its performance is better than that of the *Two-Peaks Network* as *D* can also be trained on S_1 directly, but still far from optimal. In this case, as shown in the table, the addition of the *S* module was crucial, as the gap between *SD* and *D* is much wider than before. At the same time however, with the *SD* architecture we are still able to outperform approaches relying on much more complex networks, and in particular, the one from Agresti et al. [21] (that introduced the dataset S_1), by more than 1 cm. To conclude, we will now see how our the *Direct-Global Separation Network* fares in the presence of extremely high amounts of shot noise. To this aim, we trained our *DS* architecture on the iToF2dToF dataset [1], which, as described in Section 4, has a very high amount of shot noise. To put things into perspective, the single frequency reconstruction at 100 MHz of the test set from the measurements with shot noise leads to a MAE of 7.24 cm,

while the same computation done on images with only MPI, gives an error of 1.45 cm, meaning that MPI accounts only for 20% of the total reconstruction error.

Following the setup from [1], we used two input frequencies, 20 and 100 MHz, masked the edges using a Canny edge detector during testing and did not consider the highest 1% of errors for the final computation. Our approach shows some remarkable denoising capabilities even in this scenario as it can be seen in Figure 5.10. Quantitatively, our approach reaches a test error of 1.97 cm, removing around 75% of the noise, that is behind the performances of iToF2dToF, which removes around 82% of the noise, but this is to be expected considering the very different scenario. Our approach consists of a very light architecture with extremely good MPI denoising capabilities, which is also able to deal with relatively high amount of shot noise, but the removal of zero-mean noise sources is not its primary objective, while iToF2dToF mostly focuses on this task. Moreover, the difference in complexity between the two architectures is striking: iToF2dToF needs almost two million parameters, while our network is still able to remove three quarters of the total noise using 100 times less parameters.

5.4.3 Transient Reconstruction

We will now provide some qualitative results on the performance of the *Transient Reconstruction Module* of the *Direct-Global Separation Network*, highlighting its pros and current limitations, and then make a qualitative comparison with iToF2dToF [1], the only other data-driven model that tries to reconstruct transient information. In Figure 5.11 we can see a comparison between the transient ground truth and the reconstruction of our network for 4 pixels from our transient dataset. On the top row we show a pair of good examples, where both the direct and the global components are captured quite well; on the bottom row instead we can see some of the limitations of our model. The direct component still shows a good reconstruction, while the global is more challenging to be reconstructed. The y axis has been logarithmically scaled to show both the direct and global components. In Figure 5.12 we show the performance of our approach on two pixels from the iToF2dToF dataset. Since the direct component of the transient pixel is very spread in this case, and our method only predicts a single peak, we show also an edited version of the ground truth for a better comparison. We substituted the original direct component with a single peak whose magnitude corresponds to the sum of all elements of the original direct, and whose position is taken from

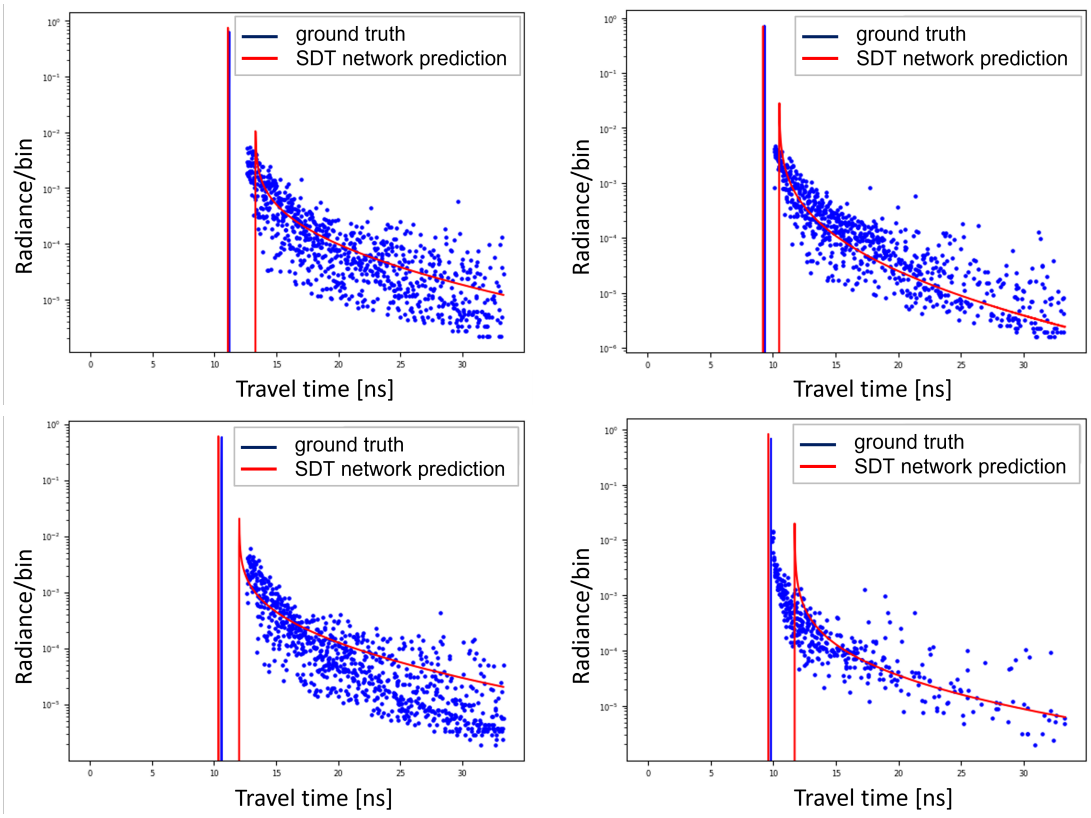


Figure 5.11: Qualitative examples showing the transient reconstruction capabilities of our approach. On the top row we show a pair of good examples, while on the bottom one a pair of less accurate ones. All the plots have a logarithmic scaling.

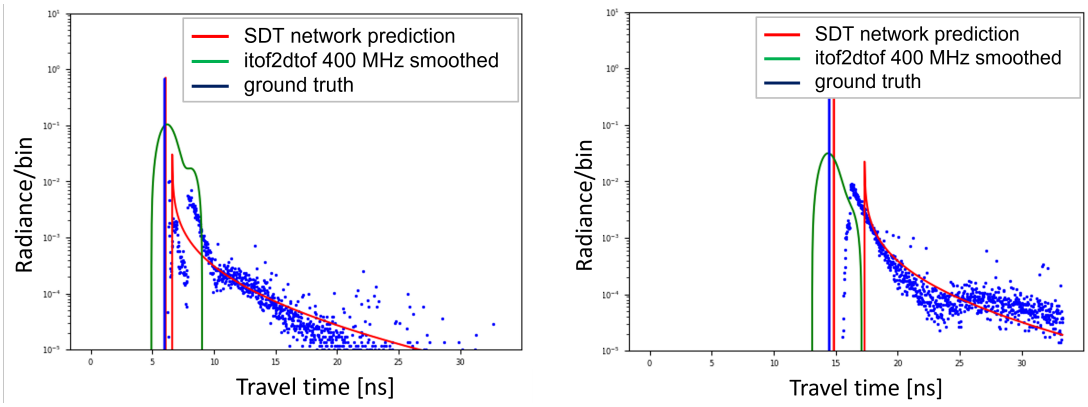


Figure 5.12: Qualitative comparison on two pixels from the iToF2dToF dataset. The direct ground truth has been substituted by a peak whose magnitude consists of the sum of the whole direct, and its position of the weighted average of the direct elements.

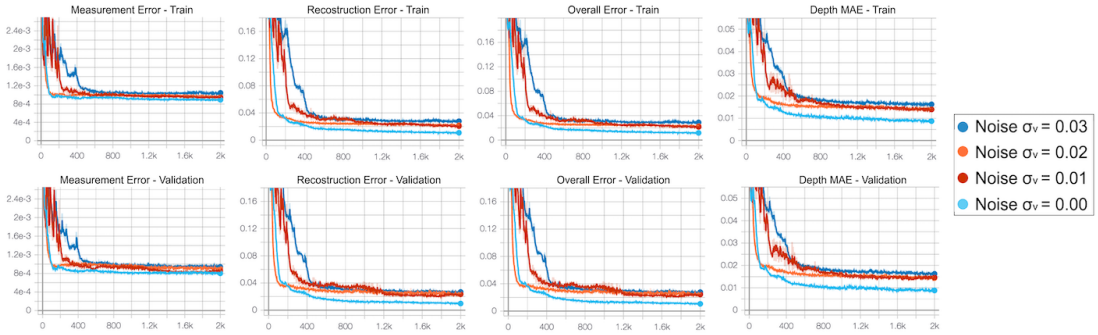


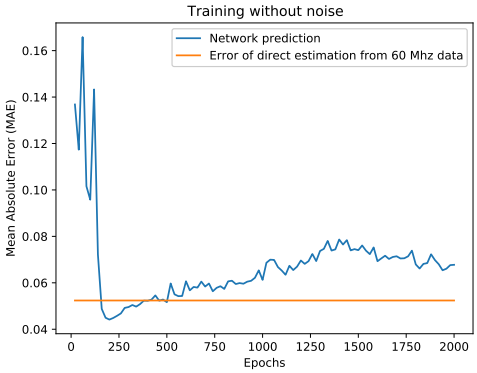
Figure 5.13: Training curves obtained running the network optimization for noise levels σ_v ={0.00;0.01;0.02;0.03} on training and validation sets. The metrics monitored are, from left to right, the measurement error, the reconstruction error, the overall error and the MAE on the depth estimated using the predicted output backscattering vector on synthetic data.

the weighted sum of all direct elements' positions, with each weight consisting of the element value itself. We can see that our method shows promising performances also on a previously unseen dataset, capturing very precisely the direct component, and reconstructing reasonably well the global. It is also clear that our model proposes a much more convincing reconstruction w.r.t. that of iToF2dToF for both components, as the competitor has a much worse estimate, especially for the global component.

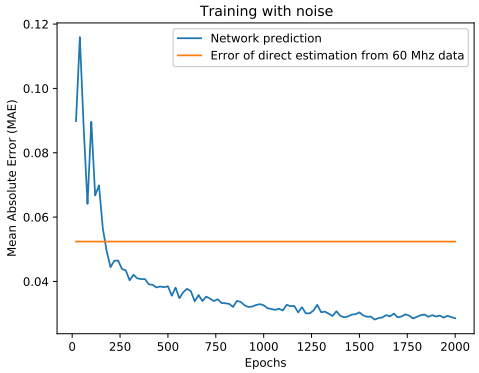
5.4.4 Ablation Studies for the Two-Peaks Network

In this section we present some ablation studies to evaluate the impact of some of the employed design choices. In particular we focus on the addition of noise during training, on the exploitation of the spatial correlation between pixels and finally on the choice of the loss function.

Firstly, we will consider the addition of noise to the simulations. As we have mentioned, there is a loss in performance when switching from synthetic to realistic data, due to the different characteristics of the two sets. The addition of noise helps the generalization capabilities of the network and reduces the gap. We repeated the training multiple times, varying the standard deviation of the noise σ_v added to the input data. At each epoch we monitored the behaviour of measurement error $\mathcal{L}_m$, reconstruction error $\mathcal{L}_r$ and overall error $\mathcal{L} = \mathcal{L}_m + \mathcal{L}_r$, as well as the MAE on the depth estimated using the predicted output backscattering vector. Figure 5.13 reports the behaviour of the considered metrics during the optimization on both the synthetic training and validation sets. As expected, the higher the noise level, the larger the errors of the predictive



(a) Training without noise ($\sigma_v = 0$)



(b) Training with noise ($\sigma_v = 0.02$)

Figure 5.14: Performance of a network trained on synthetic data with or without noise on the S3 dataset.

σ_v	0.00	0.01	0.02	0.03
MAE [cm]	4.02	2.65	**2.58**	2.83

Table 5.4: MAE on the S_3 dataset for different amounts of noise. Window size is 3×3.

Window size	1×1	3×3	5×5	7×7
MAE [cm]	2.72	**2.58**	2.61	2.80

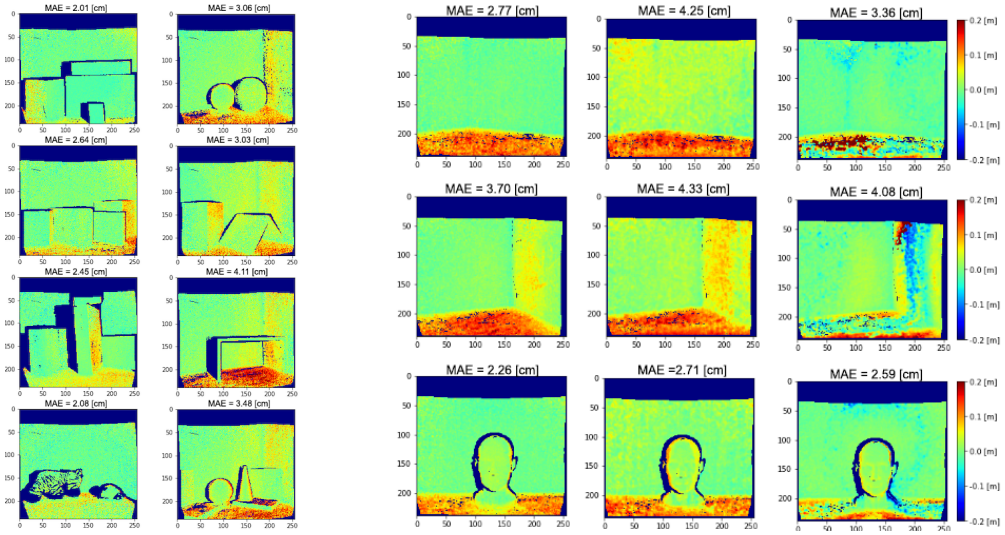
Table 5.5: MAE on the S_3 dataset for different window sizes. Noise level is $\sigma_v = 0.02$.

model will be for the synthetic data. More interesting is instead the behaviour on real data: in Figure 5.14a we show the performance of the network trained without noise on the dataset S_3 compared to a direct estimation of the depth from the component at 60 MHz. It is clear that after a promising start, the network performance starts quickly degrading as it better learns the synthetic dataset; as the trend goes on we quickly get to the point where its performance gets worse with respect even to a rough estimation based on a single frequency component. The opposite trend is instead shown in Figure 5.14b, where we display the network performance over the S_3 dataset after training the model with a noise std of $\sigma_v = 0.02$. The network quickly outperforms the naive reconstruction based on a single modulation frequency while showing an overall better behaviour.

In Table 5.4, we show the overall best performance of our network on the S_3 dataset for different noise levels. While a small amount of noise helps with the generalization, if it gets too high we instead make the task too hard to solve. Experimentally, a noise std of $\sigma_v = 0.02$ turned out to be the best compromise as it can be seen in the table.

Another line of investigation is the relevance of the spatial correlation in the final prediction. The idea is that the network should take advantage also from the local information coming from a small neighbourhood around each pixel to produce a more reliable result. To this end, we trained our network for increasing kernel sizes and evaluated its performance on the real datasets. In Table 5.5 we can see that we have the best results for a window size of 3×3 pixels around the central one, while they get worse for increasing sizes. Focusing on the network trained without spatial correlation (1×1 windows) the average MAEs obtained over S_4 and S_5 are respectively 3.43 and 2.52 cm, which turn into 2.99 and 1.88 cm after some bilateral filtering. Experimental results confirm our intuition. In the noise-free case the two networks converge to very similar results, while in the case of noise the spatial correlation helps providing a smoother prediction making the network more resilient against noise. Figure 5.15a reports the depth error maps we get on dataset S_3 without exploiting the spatial correlation.
In Figure 5.15b we show the results obtained on three real scenes for values of $P = 1, 2$ and 3 (corresponding to windows of size 3×3, 5×5 and 7×7). It is quite clear how a bigger window size generates some strong artifacts and leads to over-correcting MPI. One more time we stress the importance of linking the final prediction for each pixel to the corresponding input pixel since it is the one which carries the most information, using only a small amount of spatial correlation to refine the estimation.

The choice of a proper loss function is a crucial point in the machine learning pipeline. As already stated, some problems arise with the reconstruction loss function $\mathcal{L}_r$ since we are dealing with highly sparse vectors and in this case the gradient likely vanishes. In our study we evaluated many different loss function models in order to identify the best suitable one for our task. We started from the common MAE and MSE but they produce an all-zero output backscattering vector in most of the cases. Intuitively, looking at the behaviour of these loss functions in Figure 5.16 it is evident that applying the gradient descent algorithm the solution easily gets stuck on bad local minima. Then we shifted to cross-correlation based loss functions obtaining a significant improvement in the final prediction but they still are subjected to numerical instabilities during the optimization phase. The best results came from the weighted Earth Mover Distance introduced in Section 5.2.3 which exhibits good convergence properties and turns out to be able to drive the algorithm towards the optimal solution in a smooth fashion.



(a) **(b)**

Figure 5.15: (a) Depth error maps on dataset S_3 obtained without spatial correlation. (b) Predicted depth error maps obtained with increasing kernel size on three real scenes, from left to right respectively P=1,2 and 3.

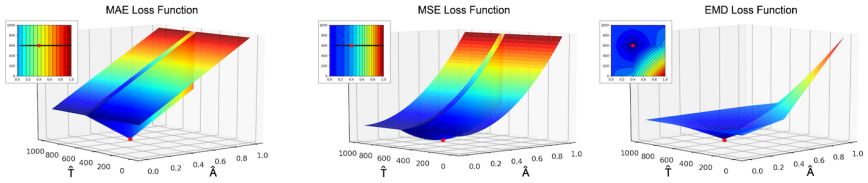


Figure 5.16: Behaviour of MAE, MSE and EMD loss functions varying amplitude $\hat{A}$ and position $\hat{T}$ of the predicted direct component.

As a final study, we decided to investigate the goodness of the prediction of our network for the intensity and time components of the two predicted peaks (corresponding to direct and global component). Since the S_3, S_4 and S_5 dataset do not have transient information, we relied on the synthetic validation dataset we introduced in Section 4 in order to compare our approach to [55], which is the only one among the compared approaches that estimates the second peak. Comparing the accuracy of the reconstruction of the global component is quite straightforward: from Table 5.2 we can retrieve the depth information (which is linked to the time displacement) and we can see that our method reaches significantly better results (2.60 and 2.12 cm on the S_4 and S_5 datasets respectively vs. 5.11 and 3.37 of [55]), while concerning the intensity values of the first peak we obtained a MAE of 0.0783 for our method against 0.1905 for SRA.

Training dataset	# of images S_1	# of images *Walls*	S_1 $[cm]$	S_4 $[cm]$	S_5 $[cm]$
S_1 (φ_d)	40	-	**5.93**	3.65	2.33
Walls (φ_d)	-	45	19.2	2.46	2.39
Walls (φ_d)	-	134	18.4	2.34	2.40
Walls (v_d)	-	134	12.2	2.26	2.44
S_1 (φ_d) + *Walls* (φ_d)	40	134	6.99	2.26	2.04
S_1 (φ_d) + *Walls* (v_d)	40	134	6.17	**2.06**	**1.87**

Table 5.6: Quantitative comparison of the performance of different training datasets on the synthetic dataset S_1 and on the real datasets S_4 and S_5 for different training datasets and losses. The evaluation metric is the MAE. All trainings have been performed on the *SD* network.

For the second peak, since it is not always present, we considered the capability of the approaches to correctly detect its presence with the well known precision-recall measures. The precision (number of pixels correctly identified as having a second peak over the total amount that have it), is 0.945 for our approach against 1 for SRA. However, this results is due to the fact that SRA overestimates the presence of the second peak as shown by the recall measure (number of pixels correctly identified as having a second peak divided by the total amount of peaks identified), with a result of 0.839 for our approach against 0.675 for SRA. Our approach also better estimates the intensity of the second peak with a MAE of 0.0702 against 0.0944 of SRA [55].

5.4.5 Ablation Studies for the Direct-Global Separation Network

As follows, we show some ablation studies regarding the *Direct-Global Separation Network* which explain the choices behind our network architecture. Among other variations, we study how the training datasets, the receptive field and the number of input frequencies influence the final performance.

TRAINING DATASET AND NUMBER OF FREQUENCIES The prediction quality of any data-driven technique heavily depends on the goodness of the dataset used to train it in the first place. For this reason, we decided to test three different scenarios: in the first one we trained our *SD* model on the S_1 dataset alone, in the second one the *Walls* dataset was the only input of the network, and in the final one we used both for supervision, as we did for the results in the previous section. In order to make the comparison fair, we also decided to perform different trainings on the *Walls* dataset

Input frequencies	S_1 dataset $[cm]$	S_4 dataset $[cm]$	S_5 dataset $[cm]$
Single frequency 50 MHz	18.0	5.31	3.82
$20, 50$ MHz	6.56	3.44	2.31
$20, 50, 60$ MHz	$\mathbf{6.17}$	$\mathbf{2.06}$	$\mathbf{1.87}$

Table 5.7: Quantitative comparison of the performance of a different number training frequencies on the synthetic dataset S_1 and on the real datasets S_4 and S_5. The evaluation metric is the MAE. The first row shows the baseline error at 50 MHz without any processing.

supervising either on $\boldsymbol{v_d}$ or on $\boldsymbol{\varphi_d}$ (i.e., the phase of the direct component). Finally, as the two datasets have a different size, we also added one entry where our method has been trained on a reduced version of our dataset (45 training images instead of 134). The outcome of this study is shown in Table 5.6. Considering first the results on S_1, we can see that the training performed on S_1 itself leads to the best performance, followed at a short distance by using both datasets; training on *Walls* alone instead falls behind by a large margin. This is not surprising as the images from *Walls* have no shot noise, thus explaining the poor performance. What's more remarkable instead is the prediction on real data, as the S_1 dataset yields a significantly poorer performance when compared to the training on *Walls*. The dissimilarity between the two datasets is striking: the former is made of much more complex scenes, shows a wide range of textures and has a good amount of simulated noise; the latter instead focuses on extremely simple structures, no changes in texture and its only noise source is MPI. What seems to be happening is that the added complexity of the dataset and some issues it presents (some scene elements of the S_1 dataset present very unreliable information), make the prediction harder for the network. Moreover, our approach relies mostly on the information in the transient direction, making in this way the complex structures from S_1 less relevant than the cleaner phasor data from *Walls*. In the end, combining the two datasets leads to the best overall solution, with a competitive performance on S_1, and the best prediction on both real datasets. It is also useful to point out that using $\boldsymbol{v_d}$ for supervision improves on training only on the phase, showing how a transient dataset can be useful for MPI correction.

Another point of interest concerns the ability of the model in dealing with a different number of input frequencies. In particular, we decided to train our model with two frequencies, 20 and 50 MHz, and see how it coped in comparison to the three frequencies

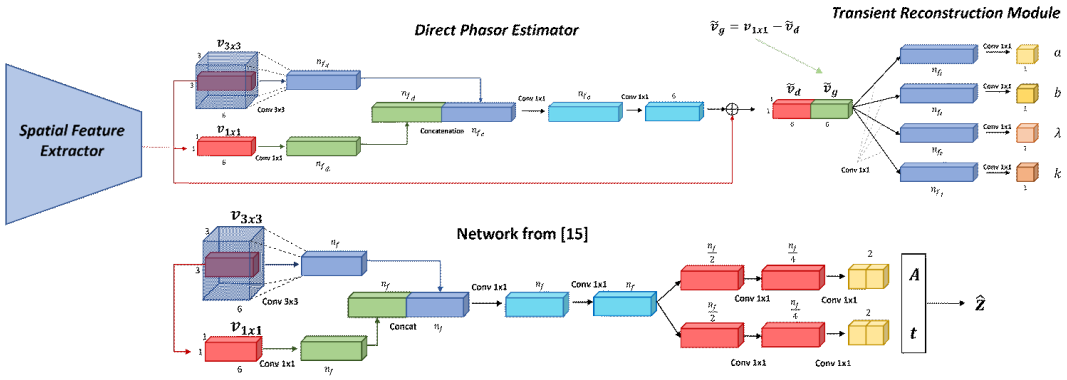


Figure 5.17: Comparison between the *Direct-Global Separation Network* and the *Two-Peaks Network*

Receptive field	S_1 dataset $[cm]$	S_4 dataset $[cm]$	S_5 dataset $[cm]$
7×7	8.08	2.44	**1.82**
11×11	**6.17**	2.06	1.87
15×15	6.35	**2.00**	2.10
21×21	8.19	2.42	2.45

Table 5.8: Quantitative comparison of the performance of different receptive fields for the *SD* network on the synthetic dataset S_1 and on the real datasets S_4 and S_5. The evaluation metric is the MAE.

input. In Table 5.7 we can see that the lack of the 60 MHz component (depth estimated from higher frequencies has typically a smaller error) has indeed a toll on the overall performance, but the model is still able to clean a noteworthy amount of MPI. To put things into perspective, it is enough to look at Table 5.6, where we can see how a 2 frequencies training on *Walls* still provides a better prediction than a 3 frequencies training on S_1 on the real datasets.

RECEPTIVE FIELD AND NETWORK COMPLEXITY As we have seen in Table 5.2, there is no clear correlation between the number of parameters of the architecture and its actual performance. This is due to multiple factors, such as the high risk of overfitting due to the domain difference between training and test data, the relatively small sizes of the datasets (even the *Walls* one only comprises around 200 images) and the main focus of the model itself, which can be centered on the use of spatial features (e.g. [21, 39]) or on the transient dimension ([22] and ours). In our case, we have to

consider an additional factor which are the characteristics of our training datasets. In particular, while a relatively large receptive field would be better in order to deal with shot noise, we cannot enlarge it too much as our main tool, the *Walls* dataset, is made mostly of flat surfaces and there is a risk of overfitting its structure. Learning too much from this dataset geometry, as it can be seen in Table 5.8, decreases the performance of the model. From the Table we can see that the overall best performance on the datasets arises from a receptive field of either 11×11 or 15×15, while it clearly degrades for smaller or bigger sizes. We decided to employ a receptive field of 11×11 due to its slightly better performance and the reduced network complexity.

5.5 Conclusions

In these chapters we have introduced a new transient dataset and two novel networks for MPI denoising and transient reconstruction. The *Walls* dataset is composed of simple structures, which however represent well enough common cases of MPI. The first method, the *Two-Peaks Network* has been built to handle high dimensional data. To do so, we split the problem in two parts: a predictive model and a backscattering model which encodes the transient information into two peaks. The network is light and reaches close to state-of-the-art performance training only on synthetic data. The *Direct-Global Subdivision Network* instead is modular: the *Spatial Feature Extractor* is deals with zero mean errors, the *Direct Phasor Estimator* deals with MPI and the *Transient Reconstruction Module* reconstructs the transient information from the previous. The *SD* architecture reaches state-of-the-art performance both on synthetic and real data, while the *D* one shows comparable performance, while only needing 3k network weights. The model shows also promising results regarding the reconstruction of transient information. A key challenge that we will explore is how to find an accurate model for the global component that can be represented with a few parameters. Moreover, we plan to substitute the parametric functions with a network in order to learn more complex global component shapes. Finally, we also plan to investigate applications of our method, such as non-line-of-sight (NLOS) imaging. It has been shown [60] that from the information carried by the global component it is possible to completely reconstruct a NLOS pixel; while the cited work relies SPAD sensors, our plan is to perform the same task starting from iToF information.

6 Depth Completion and Network Quantization

In the previous chapters we have seen, among other things, some methods for MPI denoising for iToF data. Tackling this task is particularly important as it allows to retrieve a depth image where the distortion is greatly reduced. In this chapter we will still work with iToF data, but we will consider a different setting where MPI is not an issue. In particular, we will work with an iToF sensor where the light source is not full-field, but illuminates only a set of dots as it can be seen in the first image of Figure 6.1. This procedure strongly limits the amount of light inside of the scene, thus reducing the impact of secondary light bounces to the point where MPI is negligible and the depth recovered on the illuminated areas is very accurate. This technology however has the obvious drawback of reducing the amount of reliable information as all the areas that are not illuminated have no meaningful data. In practice, if for the task of MPI correction we worked with distorted images at full resolution, in this chapter we will instead work with low resolution images where the measured depth is instead quite accurate. We will start by describing the task, which is called depth completion, together with the related literature, and then proceed to explain what network quantization is, since as before one of the objectives is to keep our architectures as light as possible. Towards the end of the chapter we will talk about existing datasets for depth completion and introduce SDS-ST1k, a very sparse dataset which is going to be one of the main benchmarks for our approach. The proposed method and the results will be shown in the next chapter.

6.1 Depth Completion

The task of depth completion consists in the estimation of a dense depth map starting from a sparse depth image and a corresponding dense RGB image. This task is particularly compelling in practice as common depth sensors such a dToF ones tend to have

a lower resolution w.r.t. common RGB ones. This is due to multiple factors, such as a limited fill-factor linked to pixel complexity and/or the use of a dot pattern light to limit the power consumption. In Figure 6.1, we can see an example of the inputs and outputs related to the task of depth completion.

Figure 6.1: The task of depth completion

6.1.1 Related Works

The task of depth completion has been popular in computer vision since the introduction of depth sensors [61, 62] and has been commonly addressed by neural networks with convolutional encoder-decoder architectures [63, 64, 2, 65, 4, 66, 5]. Many other works define the state of the art in depth completion [67, 68, 69, 70, 71, 72, 73, 74]. We will compare our results against CNN solutions [3, 2, 4, 5] selected on the basis of their similar complexity (single encoder-decoder). These will be retrained on our sparse ToF datasets. There exist also higher-complexity networks leveraging, *e.g.*, multiple decoder branches [75, 65] or separate encoder branches [66, 76, 77, 78] but in this work we aimed for a more compact and lightweight model. We will only focus on *single-frame* depth completion. Many contributions extend depth completion by either self-supervision or by providing as input multiple RGB-D frames with known or inferred poses [2, 79, 80, 78, 81]. By tackling only the single-frame case, we grant temporally-independent behavior for both dynamic and static cameras and remove additional computational effort required by pose estimation via neural networks (*e.g.*, [82]) or traditional odometry. We now discuss the main works we compare against.

SPARSE-TO-DENSE (S2D [2]) is a popular approach using a single encoder-decoder network. Its core contribution is self-supervision by pose estimation (a feature we do

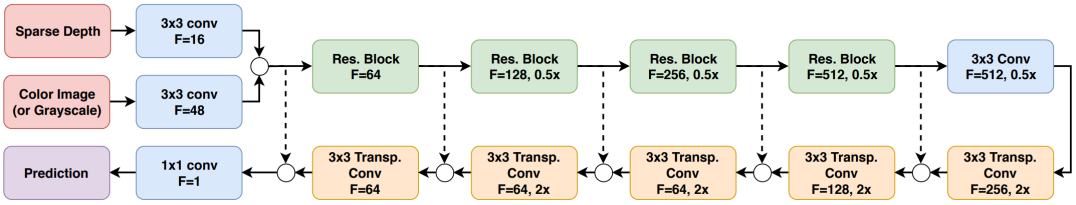


Figure 6.2: S2D Network architecture [2]

not leverage) as well as its long-standing role in the KITTI leaderboard compared to its low complexity. In its supervised flavor, the paper uses a mixed loss, with ℓ_2-loss as well as ℓ_1-loss on the prediction gradients to enforce piecewise-smoothness of the depth map. The network structure can be seen in Figure 6.2 The network architecture is similar to ours, except for the use of transpose convolution layers in the decoder, and for the use of two separate layers to pre-process the RGB and sparse depth inputs. As shown in [2, Sec. 6.4] and confirmed in our experiments, its robustness to sparsity is limited. We reproduced and retrained their model.

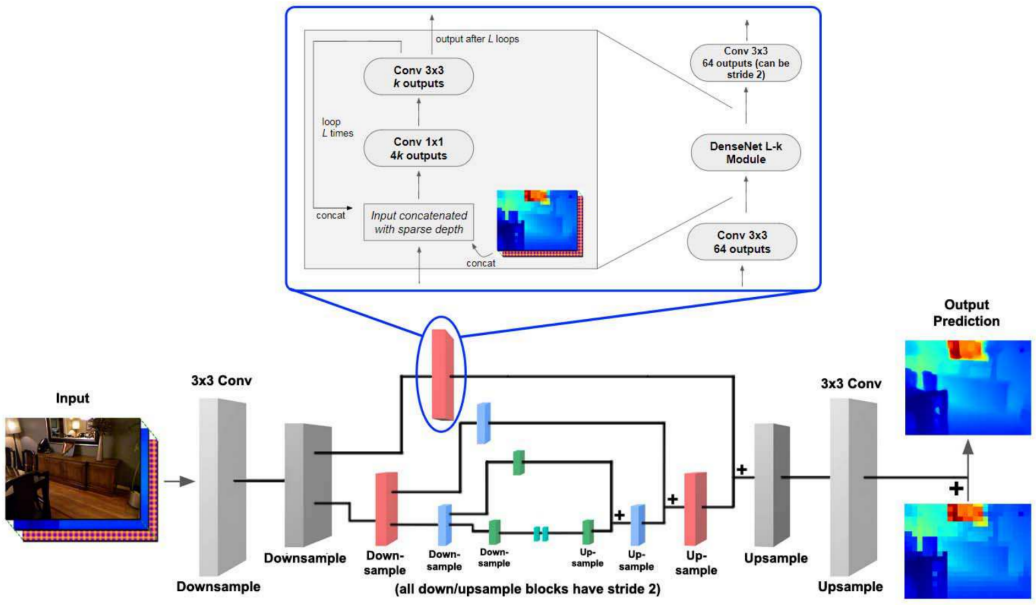


Figure 6.3: D^3 network architecture [3]

$\mathbf{D^3}$ [3] is developed using DenseNet [83] layers as building blocks. Its input pre-processing is an efficient way to initialize the depth map and attain high quality while maintaining low complexity. NYU-Depth v2 results are provided, but the authors' code is not public and we could not reproduce their quality in our implementation and retraining.

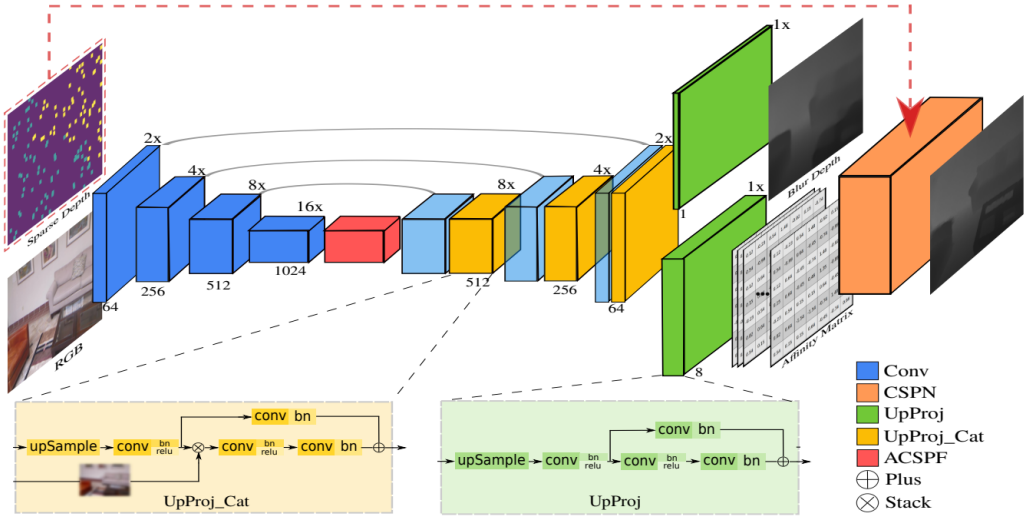


Figure 6.4: CSPN network architecture [4]

CONVOLUTIONAL SPATIAL PROPAGATION NETWORK (CSPN [4]) was among the first works [4, 75, 5] to propose a two-stage network architecture using *spatial propagation* layers. This entails: (i) generation of a "blurred" depth map and affinity matrix using an encoder-decoder network ; (ii) iterative refinement using the learned affinity matrix as weights for anisotropic diffusion. We retrained the authors' model via their code.

NON-LOCAL SPATIAL PROPAGATION NETWORK (NLSPN [5]) improves upon [4], its main novelty being the use of non-local neighbors in the diffusion process, which is implemented by deformable convolutions. The authors show it achieves superior performances w.r.t. S2D and CSPN on NYU-Depth v2. In our experiments, we confirm this is indeed the most competitive approach at its network complexity. Further improvements were recently presented [84]. We retrained the authors' model via their code.

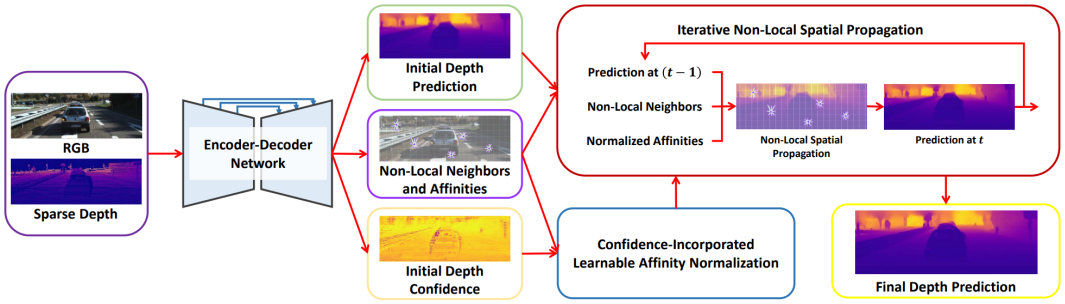


Figure 6.5: NLSPN network architecture [5]

6.1.2 Neural Networks Quantization

Quantized Neural Networks (QNN) [85, 86, 87] are DNNs for which a smaller number of bits $b \ll 32$ is used to represent the weights and activations. QNNs are key to achieving efficient and low-power inference: they require considerably less memory and have a lower computational complexity, since quantized values can be stored, multiplied and accumulated efficiently.

Quantization-Aware Training (QAT) refers to a set of special algorithms to train QNNs, leading to superior results than post-training quantization. Recent QAT methods [88, 89, 90, 91, 92] proved effective on classification tasks.

These methods often use uniform quantization and the same bit width in each network layer. To improve QAT, Uhlich *et al.* [93] and Nikolić *et al.* [94] proposed to *learn* the optimal bit width for each layer while achieving a target size budget. On classification tasks, these *mixed precision* approaches show superior results since the bit widths can be allocated optimally across the network. Much less evidence of successful QAT exists for regression tasks such as depth completion.

6.2 Sparse ToF Datasets

We target datasets that are critically challenging in terms of sparsity. We consider indoor scenes (range: $[0, 15]m$) with rich structure and detailed objects with diverse camera orientations. Given sparse depth maps D_{S}, let us define the *sparsity level* $K := 100\frac{|\{i \in [n_h] \times [n_v] : (D_{\mathrm{S}})_i \neq \mathrm{Invalid}\}|}{n_h n_v}$ at target depth map resolution* $n_h \times n_v$. We tackle very

*Typically equal to the input RGB frame resolution or a cropping of it.

(a) $1216 \times 352, K \approx 4.4\%$

(b) $304 \times 224, K \approx 1.4\%$

(c) $640 \times 480, K \approx 0.4\%$

Figure 6.6: RGB-annotation overlay of (a) KITTI (LiDAR; range: $[0, 85]m$), (b) NYU-Depth v2 (processed; range: $[0, 10]m$), (c) SDS-ST1k (sparse ToF; range: $[0, 15]m$). Annotations (red) at available sparse depth coordinates. Figure best viewed in color.

sparse settings where $K \approx 0.4 \sim 1.4\%$ (as a reference in the widely used KITTI depth completion benchmark[†] $K \approx 4 \sim 5\%$). A visual overview is provided in Fig. 6.6.

Indeed, such higher values of K simplify the depth completion task. In this work we will focus on indoor 3D perception datasets with very sparse depth maps, where traditional pipelines [95] will struggle to achieve good results.

Our main results are given on NYU-Depth v2 [96] as broadly adopted public-domain dataset. For the training set, we use a subset of $\approx$ 50K RGB-D images from the 249 standard training scenes. Each image is downsized to 320×240 then center-cropped to 304×224, identically to [63, 5]. The main difference we introduce is that we process the depth by subsampling it with a triangular tiling dot pattern generated from sparse illumination of commercial VCSELs [97] instead of choosing random indices. This results in sparse depth maps D_{S} with on-average 943 active pixels ($K \approx 1.4\%$; see Fig. 6.6b) sampled from the full, dense depth map D_{GT} which is used as supervision.

[†]By computing K over the standard validation set of KITTI Depth Completion.

The surface normals map N_{GT} was also estimated from D_{GT}. The standard test set of 654 images is processed identically.

In addition, we provide results on SDS-ST1k [98], a contributed dataset using 8 diverse environments from Unreal Engine [99] marketplace assets yielding $\approx$ $18\mathrm{K}$ images from random camera poses at resolution 640×480. To obtain sparse ToF data, we apply a light source with dot pattern light shading in our raytracing ToF simulator; this is received on a simulated ToF sensor compatible with low power designs and camera specifications (*e.g.*, sensor integration time, camera intrinsics). This provides both sparse ToF depth maps D_{s} with realistic parallax and pattern geometry, and ground truth $D_{\mathrm{GT}}, N_{\mathrm{GT}}$. In this setting we obtain on-average 1239 active pixels out of 640×480 depth map values, , yielding a challenging $K \approx 0.4\%$ (see Fig. 6.6c). The recommended training/testing split yields $\approx 5\mathrm{K}$ images as test set; this will be provided at `https://github.com/sony/ai-research-code`. We shall train our models with 160×160 non-overlapping patches (*i.e.*, 12 per image) to meet training device memory limitations. We also augment the training set by random horizontal and vertical flipping. We generate a mask as hexagonal sparsity pattern and 943 depth pixels are sampled from the dense depth map, producing the sparsity level of 1.385%.

7 Quantized Deep learning Approach for Depth Completion

In this chapter, we will provide an in-depth description of the approach we take for depth completion. We start by describing the preprocessing of our data and the network architecture we start from; we then explain all the task-related improvements we introduce and the loss function we employed. Finally, we explain the quantization approaches we applied to reduce the network size and conclude with the experimental results.

7.1 Network Design

The input of all considered depth completion methods is an RGB-D frame comprised of a colour image and a sparse depth map, suitably projected to RGB camera space by assuming known intrinsics and RGB-ToF extrinsics, and with negligible RGB-ToF baseline to avoid occlusions.

7.1.1 Input Pre-Processing

We leverage the input pre-processing method of Chen *et al.* [3]: given the sparse depth D_{S}, we compute an initial guess D_{NNI} of the depth map by Nearest Neighbors Interpolation (NNI), *i.e.*, mapping each coordinate to the closest sparse depth pixel in the Euclidean sense. We also compute the Euclidean Distance Transform (EDT) E with [100] (via OpenCV) given the 2-D valid coordinates of D_{S}. NNI is obtained by the same call, which takes median CPU time $t_{\mathrm{NNI}} = 4.99\ ms$ over SDS-ST1k and $t_{\mathrm{NNI}} = 0.99\ ms$ over NYU-Depth v2. We normalize D_{NNI} by the maximum range $D_{\max} := 15m$ and E by $E_{\max} := 40$, which we find empirically to be an appropriate normalization value for our datasets. The EDT feeds the CNN an uncertainty map of D_{NNI}, *i.e.*, it is low where sparse depth is known and progressively higher when it is

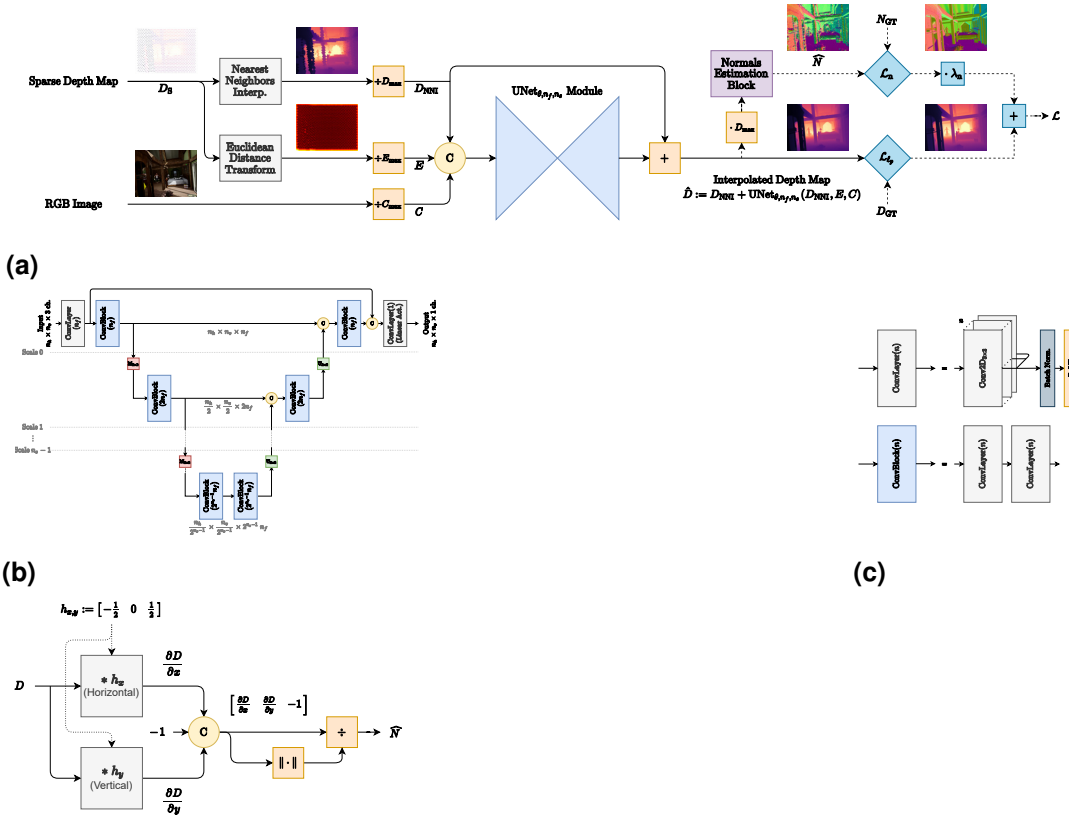


Figure 7.1: Summary of our network design for depth completion: (a) Network architecture. We highlight pre-processing blocks (gray); tensor operators (orange, square); training supervision operators (teal) and connections (dashed); concatenation (yellow, circle); the UNet$_{\theta,n_f,n_s}$ module (blue); the normals estimation block (purple). (b) UNet$_{\theta,n_f,n_s}$ Module. $\mathrm{M}_{2\times2}$ indicates a MaxPooling2D layer with stride 2. $\mathrm{U}_{2\times2}$ indicates an Upsample2D layer using nearest neighbors upsampling with factor 2. (c) Basic convolutional building blocks of the UNet$_{\theta,n_f,n_s}$ Module. (d) Normals Estimation Block, as implemented by 1-D convolution ($*$) in the respective directions.

not. We tried replacing NNI with linear interpolation* in preliminary tests, but this did not show any benefits while increasing CPU time.

7.1.2 Network Architecture

A summary of our supervised approach is given in Fig. 7.1. The core CNN element (Fig. 7.1b) is a simple "UNet-like" [101] template, dubbed UNet$_{\theta,n_f,n_s}$, of which we

*Linear interpolation on a non-uniform grid due to the dot pattern requires Delaunay triangulation, which is costly and cannot be reused as the pattern can change at every frame.

tune the number of scales n_s and feature maps n_f relative to the highest scale. Our reference is $n_s = 5, n_f = 64$. The number of feature maps doubles at each scale as their spatial resolution is reduced, up to 2^{n_s-1} at the lowest scale. The basic building blocks (Fig. 7.1c) are standard 2-D convolutions with 3×3 filters, with padding to same resolution as the layer input and stride 1. We chose to replace transpose convolutions in the decoder with upsampling and convolution layers, as preliminary tests indicated better performance on our datasets. All skip connections are by concatenation except for the last one, which is a tensor addition w.r.t. D_{NNI}. Thus, we have the (normalized) interpolated depth $\hat{D} := D_{\text{NNI}} + \text{UNet}_{\theta,n_f,n_s}(D_{\text{NNI}}, E, C)$, *i.e.*, $\text{UNet}_{\theta,n_f,n_s}$ computes a residual w.r.t. the initial guess.

7.1.3 Normals Estimation Block

As shown in Fig. 7.1a, we estimate the surface normals map $\hat{N}$ from the interpolated depth map $\hat{D}$ rather than using costly dedicated decoders [65]. We resort to a well-known approximation that is directly applicable to depth maps; the computation graph is reported in Fig. 7.1d. Given a depth map tensor D at the input, we approximate horizontal and vertical image axes derivatives by centered differences using two fixed-kernel 1-D convolutions yielding the tensors $(\frac{\partial D}{\partial x}, \frac{\partial D}{\partial y})$. We then have at the i-th pixel the normal vector

$$(\hat{N})_i := \frac{\left[(\frac{\partial D}{\partial x})_i \quad (\frac{\partial D}{\partial y})_i \quad -1\right]}{\sqrt{(\frac{\partial D}{\partial x})_i^2+(\frac{\partial D}{\partial y})_i^2+1}}. \tag{7.1}$$

We found this small-kernel method sufficiently accurate to enforce similarity[†] between $(\hat{N})_i, (N_{\text{GT}})_i$ at training.

7.1.4 Loss Function

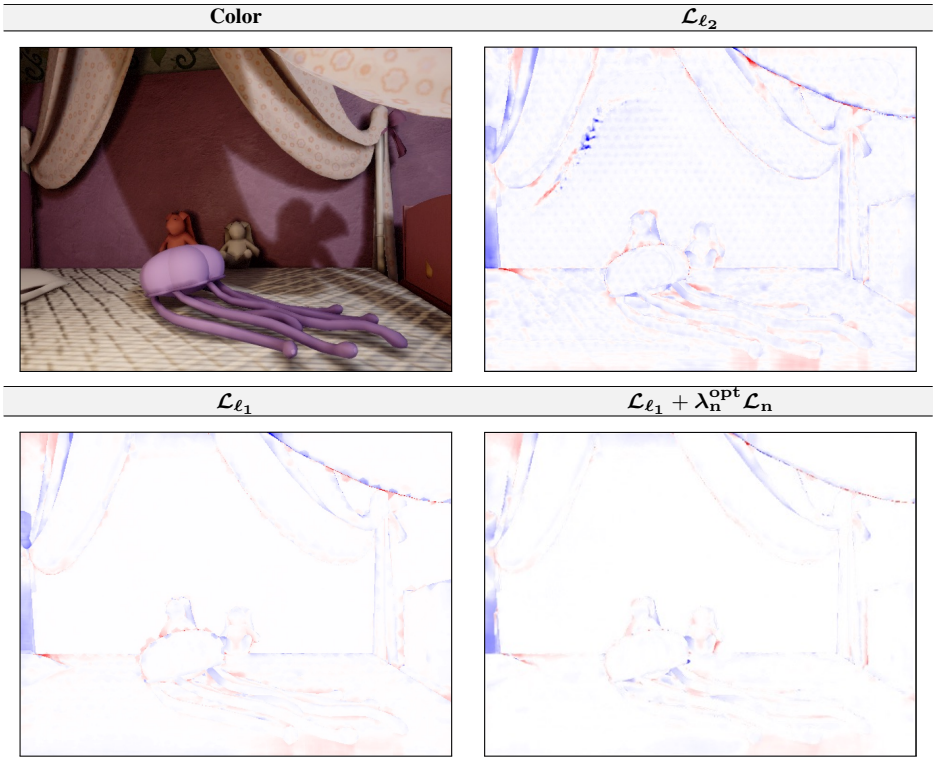
Our supervised training follows the flow in Fig. 7.1a. The supervision is comprised of two terms. The first term measures the ℓ_p-norm distance between (normalized) ground truth D_{GT} and $\hat{D}$, *i.e.*, for dataset $\mathcal{D}$ with J samples,

$$\mathcal{L}_{\ell_p}(\mathcal{D}) := \tfrac{1}{J}\textstyle\sum_{j=1}^{J} \| \operatorname{vec}(\hat{D}_j - D_{\text{GT},j})\|_p^p, \; p = 1, 2. \tag{7.2}$$

[†]We ensure the surface normals' orientation is always consistently pointing towards the camera both in N_{GT} and $\hat{N}$.

$\mathcal{L}$	λ_{n}	RMSE (mm)	MNS
$\mathcal{L}_{\ell_2}$	n/a	105.2	0.538
$\mathcal{L}_{\ell_1}$	n/a	97.8	0.754
$\mathcal{L}_{\ell_1} + \lambda_{\mathrm{n}}\mathcal{L}_{\mathrm{n}}$	1	259.0	0.864
$\mathcal{L}_{\ell_1} + \lambda_{\mathrm{n}}\mathcal{L}_{\mathrm{n}}$	10^{-1}	227.0	0.868
$\mathcal{L}_{\ell_1} + \lambda_{\mathrm{n}}\mathcal{L}_{\mathrm{n}}$	10^{-2}	101.5	0.838
$\mathcal{L}_{\ell_1} + \lambda_{\mathrm{n}}\mathcal{L}_{\mathrm{n}}$	10^{-3}	98.7	0.783
$\mathcal{L}_{\ell_1} + \lambda_{\mathrm{n}}\mathcal{L}_{\mathrm{n}}$	10^{-4}	109.3	0.764

(a)



(b)

Figure 7.2: Loss function tuning on SDS-ST1k (resolution 640×480): (a) Quantitative analysis; (b) Qualitative analysis of error maps (range: $[-500, 500]\,mm$). Figure best viewed in color.

The second term measures similarity between the dense ground truth normals map N_{GT} and $\hat{N}$ from the block in Fig. 7.1d: for each i-th pixel normal we measure the cosine similarity $\cos\alpha = \langle (N_{\mathrm{GT}})_i, (\hat{N})_i \rangle$. We then take its average over the dataset and define the normals loss

$$\mathcal{L}_{\mathrm{n}}(\mathcal{D}) := -\frac{1}{Jn_h n_v} \sum_{j=1}^{J} \sum_{i=1}^{n_h n_v} \langle (N_{\mathrm{GT},j})_i, (\hat{N}_j)_i \rangle \tag{7.3}$$

with range $[-1, 1]$ (at -1 all normals are identical). In the presence of invalid ground truth depth or normals map (*e.g.*, for invalid pixels in NYU-Depth v2) we exclude the corresponding pixel from the loss computations.

Only part of the literature uses normals supervision[‡] [65, 75], while all depth completion approaches use depth supervision providing scale. Normals supervision alone is not sufficient to retrieve scale. Thus, we minimize

$$\mathcal{L}(\mathcal{D}) := \mathcal{L}_{\ell_p}(\mathcal{D}) + \lambda_{\mathrm{n}} \mathcal{L}_{\mathrm{n}}(\mathcal{D}) \tag{7.4}$$

which balances between scale-dependent and scale-independent terms using $\lambda_{\mathrm{n}} \in [0, 1]$. This weight depends on the dataset and must be tuned empirically. To do so, we run preliminary tests of our approach (Fig. 7.1a) on SDS-ST1k under different[§] $\mathcal{L} := \{\mathcal{L}_{\ell_2}, \mathcal{L}_{\ell_1}, \mathcal{L}_{\ell_1} + \lambda_{\mathrm{n}} \mathcal{L}_{\mathrm{n}}\}$ and $\lambda_{\mathrm{n}} := 10^{-q}, q = 0, \ldots, 4$, in the same settings described in Sec. 7.3.1. Indeed, defining the conventional RMSE and Mean Normals Similarity

$$\mathrm{MNS} := \frac{1}{Jn_h n_v} \sum_{j=1}^{J} \sum_{i=1}^{n_h n_v} \langle (N_{\mathrm{GT},j})_i, (\hat{N}_j)_i \rangle \tag{7.5}$$

as quality metrics we see that large values of λ_{n} degrade the RMSE, and *vice versa* for MNS. We find the best trade-off at $\lambda_{\mathrm{n}}^{\mathrm{opt}} = 10^{-3}$. As confirmation of this, in Fig. 7.2b we report one sample error map $\hat{D} - D_{\mathrm{GT}}$ under different $\mathcal{L}$. As we move from $\mathcal{L}_{\ell_2}$ to $\mathcal{L}_{\ell_1}$ the observed error map pattern artefact disappears. Remaining errors at depth map discontinuities are reduced by using the normals loss $\mathcal{L}_{\mathrm{n}}$.

n_f	n_s	MParams	RMSE (mm)	MNS
64	5	50.3	98.7	0.783
32	5	12.6	105.0	0.775
16	5	3.2	181.1	0.751
64	4	12.6	219.2	0.779
64	3	3.1	248.4	0.761
32	4	3.1	274.8	0.767

Table 7.1: Network tuning on SDS-ST1k (resolution 640×480). We confirm a steady decrease of quality metrics as the feature maps n_f relative to the highest scale and the scales n_s are reduced.

7.1.5 Network Tuning

We investigated simple strategies to reduce the size of the initial model, such as tuning the $\text{UNet}_{\theta,n_f,n_s}$ module by its number of feature maps n_f relative to the highest scale, and the number of scales n_s. Our preliminary tests, in the same setting as Sec. 7.3.1 on SDS-ST1k, actually confirm (Tab. 7.1) that any pair smaller than $(n_f, n_s) = (64, 5)$ leads to a rapid degradation of RMSE and MNS, with the reduction of n_s (shallower network) being generally worse than that of n_f. Thus, we choose to achieve memory size reduction by quantization of the full model with $(n_f, n_s) = (64, 5)$ which, as we will show, allows for a more compact model with limited impact on quality metrics.

7.2 Quantization-Aware Training

Let $Q(x; \boldsymbol{\phi})$ be an arbitrary scalar quantizer with parameters $\boldsymbol{\phi}$, which maps $x \in \mathbb{R}$ to discrete values $\{q_1, ..., q_I\}$ represented by b bits, $b : I \leq 2^b$. This quantizer may be applied to both weights and activations to reduce their memory size requirements. This can be done by either post-training quantization or by Quantization-Aware Training (QAT), which generally achieves better accuracy than the former. QAT consists in

[‡]Smoothness losses on the gradients of $\hat{D}$ are also common [2, 77] but not considered here as the normals loss implements similar behavior in a supervised and more geometrically sound fashion.

[§]We verified in preliminary tests that $\mathcal{L}_{\ell_2} + \lambda_{\text{n}}\mathcal{L}_{\text{n}}$ is systematically worse than the ℓ_1 case and excluded it accordingly.

training the quantized version of a DNN (in this case, CNN) while applying $Q(x;\boldsymbol{\phi})$ to its weights and/or activations. However, this is challenging as the gradient through the quantizer is zero almost everywhere; a solution to this is the Straight-Through Estimator (STE) [102]. A *symmetric uniform* quantizer $Q_U(x;\boldsymbol{\phi})$ then maps $x \in \mathbb{R}$ to uniformly quantized values by

$$q = Q_U(x;\boldsymbol{\phi}) := \text{sign}(x) \begin{cases} d \left\lfloor \frac{|x|}{d} + \frac{1}{2} \right\rfloor & |x| \leq q_{\max} \\ q_{\max} & |x| > q_{\max} \end{cases}, \tag{7.6}$$

where the parameter vector $\boldsymbol{\phi} := [d, q_{\max}, b]^T$ with step size $d \in \mathbb{R}^+$, maximum value $q_{\max} \in \mathbb{R}^+$, and number of bits or *bit width* $b \in \mathbb{N} : b \geq 2$ of the quantized value q.

7.2.1 Uniform Precision Models

We start by considering *uniform precision* (UP) QAT, *i.e.*, every weight (or activation) of every layer has the same predefined bit width b and QAT optimizes over the remaining parameters $d, q_{\max}$ of $\boldsymbol{\phi}$. We conducted extensive experiments to preserve depth completion accuracy while quantizing with (7.6) the weights and activations in our model. The most effective UP QAT training procedure was obtained by initializing from the pretrained $\text{float}32$ model and using learnable $q_{\max}$ as in the Trained Quantization Threshold (TQT) approach [92]; to stabilize the QAT, we used cosine learning rate decay scheduling [103], and we first trained with RMSprop until convergence (32 epochs in our case), followed by Adam [104] for 20 more epochs.

7.2.2 Mixed Precision Models

Recent results [93, 94] show that, for a given memory size budget, better accuracy than UP QAT can be achieved using *mixed precision* (MP) QAT, *i.e.*, each layer uses its own bit width *learned at training* to fit a target total network size. Following [93] we parametrize $Q_U(x;\boldsymbol{\phi})$ with their range and step size d, from which the bit width $b := \lceil 1 + \log_2(q_{\max}/d + 1) \rceil$ is then inferred. With this parametrization and STE [102] gradient, we can use standard stochastic gradient descent methods to learn per-layer optimal bit widths jointly with network parameters. To achieve target network sizes we add to (7.4) *size constraints* on the weights and activations as penalty terms [105],

minimizing

$$\mathcal{L}_{\mathrm{MP}} = \mathcal{L}(\mathcal{D}) + \lambda_{\mathrm{W}} \max(0, S_{\mathrm{W}})^2 + \lambda_{\mathrm{A}} \max(0, S_{\mathrm{A}})^2 \tag{7.7}$$

$$S_x = (\textstyle\sum_{l=1}^{L} S_x^l) - S_x^o, S_x^l = N_{x,l} b_{x,l}, S_x^o = \bar{b}_x \textstyle\sum_{l=1}^{L} N_{x,l} \tag{7.8}$$

where $x := \{\mathrm{W}, \mathrm{A}\}$ denote weights or activations; S_x^l their per-layer total memory size for $N_{x,l}$ coefficients with learned bit widths $b_{x,l}$; S_x^o is the *target size* which we tune[¶] by a reference *average bit width* $\bar{b}_x$. The parameters $\lambda_{\mathrm{W}} \in \mathbb{R}^+$ and $\lambda_{\mathrm{A}} \in \mathbb{R}^+$ are chosen to balance the respective penalties. We empirically set them following the criteria in [93, Sec. 4], yielding $\lambda_{\mathrm{W}} \approx 2.66 \cdot 10^{-7}, \lambda_{\mathrm{A}} \approx 1.73 \cdot 10^{-6}$. In MP QAT we also initialize network parameters from the pretrained $\mathrm{float32}$ model, and run Adam for 60 epochs until convergence with cosine learning rate decay scheduling.

7.2.3 Weights Quantization

We initially compare *weights-only* UP QAT and MP QAT against the pretrained $\mathrm{float32}$ model, with the same dataset and setting as Sec. 7.1.5. In all our models the last layer is at $\mathrm{float32}$ precision as it directly produces the regression output in Fig. 7.1a. We refer to UP models $\mathrm{W}_b^{\mathrm{UP}}$ and MP models W_b at (fixed or average) bit width b. We report the most compact UP model $\mathrm{W}_4^{\mathrm{UP}}$ ($b = 4$) providing best quality metrics, and compare it against the most compact MP model $\mathrm{W}_\star$, where $\star$ denotes target size $S_{\mathrm{W},\star}^o := 14\,\mathrm{MB}$, *i.e.*, $\bar{b}_{\mathrm{W}} \approx 2.35\,\mathrm{bit}$ when setting the size constraint.

The results in Fig. 7.3a allow us to establish that MP QAT is superior to UP, as with half of the weights memory size we see no degradation of MNS and a slight improvement in RMSE due to refinement from the pretrained $\mathrm{float32}$ model. We also observe in Fig. 7.3b how the per-layer precision allocation of MP naturally tends to assign smaller (larger) bit widths to inner (outer) layers of the QNN.

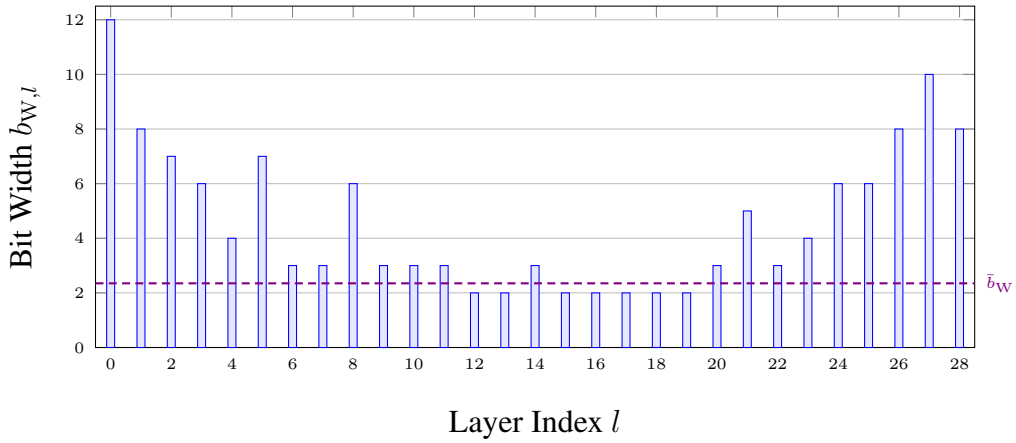
7.2.4 Activations Quantization

For efficient inference on mobile or low-power devices, it is crucial to consider the quantization of both weights and activations. Indeed, activations can dominate memory requirements at inference (especially for encoder-decoder networks such as ours) and

[¶]This tuning can arbitrarily refer to an average bit width or to a total size (in which case the average as defined in (7.8) can be fractional).

Precision	Weights Size (MB)	RMSE (mm)	MNS
float32	198.5	98.7	0.783
Uniform ($\mathrm{W}_4^{\mathrm{UP}}$)	25.1	100.3	0.762
Mixed ($\mathrm{W}_\star$)	**13.9**	**94.5**	**0.783**

(a)



(b)

Figure 7.3: Weights quantization. (a) Results on SDS-ST1k (resolution 640×480) for the most compact UP model W_4^{UP} at bit width $b = 4$ and MP model $W_\star$ at average bit width $\bar{b}_{\mathrm{W}} = 2.35$. (b) Per-layer precision allocation of $\mathrm{W}_\star$ (last $\mathrm{float32}$ layer omitted) and the corresponding $\bar{b}_{\mathrm{W}}$ (dashed).

Method	Weights Size (MB)	Activation Size (MB)	RMSE (mm)	MAE (mm)	MRE (%)	MNS	$\boldsymbol{\delta}_1$(%)	$\boldsymbol{\delta}_2$(%)	$\boldsymbol{\delta}_3$(%)	Median t_{GPU} (ms)
NYU-Depth v2 (center cropped, 304×224**)**										
S2D	90.1	322.7	72.1	34.1	1.230	0.538	99.79	99.97	99.99	**4.83** (0)
D^3	7.8	59.9	128.3	46.9	1.760	0.422	99.13	99.79	99.94	13.78 (0.99^{CPU})
CSPN	832.1	44.2	79.0	28.5	0.970	0.752	99.68	99.95	99.99	63.97 (14.02)
NLSPN	100.1	200.2	**61.4**	**22.5**	**0.750**	**0.800**	**99.83**	**99.98**	**100.00**	42.50 (7.93)
Ours	198.5	145.5	63.7	23.2	0.782	0.790	99.81	99.98	99.99	5.11 (0.99^{CPU})
SDS-ST1k (640×480)										
S2D	90.1	1455.5	643.3	589.9	29.130	0.158	55.45	83.27	94.25	**6.02** (0)
D^3	7.8	270.2	176.7	57.4	2.500	0.572	98.40	99.40	99.70	18.83 (4.99^{CPU})
CSPN	832.1	199.2	174.3	97.1	3.850	0.687	98.60	99.60	99.78	154.62 (23.50)
NLSPN	100.1	890.6	99.1	29.4	1.150	0.775	99.39	**99.81**	**99.92**	148.68 (11.18)
Ours	198.5	656.3	**98.7**	**21.7**	**0.829**	**0.783**	**99.45**	99.81	99.91	10.30 (4.99^{CPU})

Table 7.2: Depth completion models. The median t_{GPU} is given as $t_{\mathrm{total}}(t_{\mathrm{initialization}})$. In ours and D^3, initialization is computed on $^{\mathrm{CPU}}$.

Precision	Weights Size (MB)	Activation Size (MB)	RMSE (mm)	MAE (mm)	MRE (%)	MNS	$\boldsymbol{\delta}_1(\%)$	$\boldsymbol{\delta}_2(\%)$	$\boldsymbol{\delta}_3(\%)$
NYU-Depth v2 (center cropped, 304×224**)**									
float32	198.5	145.5	63.7	23.2	0.782	0.790	99.81	99.98	99.99
$\mathrm{W}_8\mathrm{A}_8$	22.1	47.9	**64.2**	**23.4**	**0.784**	**0.793**	**99.81**	**99.98**	**99.99**
$\mathrm{W}_4\mathrm{A}_8$	22.0	36.2	64.3	23.5	0.793	0.790	99.81	99.98	99.99
$\mathrm{W}_4\mathrm{A}_4$	23.7	**17.6**	70.3	26.1	0.894	0.739	99.76	99.97	99.95
$\mathrm{W}_\star$	**13.7**	145.5	64.9	23.6	0.796	0.786	99.81	99.98	99.99
SDS-ST1k (640×480)									
float32	198.5	656.3	98.7	21.7	0.829	0.783	99.45	99.81	99.91
$\mathrm{W}_8\mathrm{A}_8$	45.9	164.1	101.7	22.5	0.874	0.781	99.40	99.79	99.91
$\mathrm{W}_4\mathrm{A}_8$	20.7	163.3	101.3	21.1	0.810	0.776	99.48	99.83	99.92
$\mathrm{W}_4\mathrm{A}_4$	21.1	**78.4**	99.0	24.4	0.958	0.638	99.41	99.81	99.92
$\mathrm{W}_\star$	**13.9**	656.3	**94.5**	**20.8**	**0.808**	**0.784**	**99.48**	**99.83**	**99.92**

Table 7.3: Quantized network models. Among MP QNN models, we highlight the best trade-off between lowest memory footprint and best quality metrics for weights and activations (yellow) and weights-only (orange). Figure best viewed in color.

may additionally cost many read/write accesses of buffer memory, with large impact on energy consumption. As we observed that MP is capable of achieving superior performances than UP for the weights-only case, our final results (Sec. 7.3.3) will consider MP QAT with weights and activations quantization at 4 to 8 bits; we therefore denote them by $\mathrm{W}_{b'}\mathrm{A}_{b''}$ where $b', b'' \geq 2$ are average bit widths that set the respective size constraints.

7.3 Results

We present our results using the same quality metrics of most prior works [63, 3]: RMSE, Mean Absolute Error (MAE), Mean Absolute Relative Error (MRE), and $\delta_i, i = 1, 2, 3$; we also add the MNS metric (7.5) to study normals similarity. Where reported, the median GPU time per prediction (t_{GPU}) is measured on an NVidia GTX 1080 Ti. The reported results are given on full-frame test sets from Sec. 6.2.

7.3.1 Training Setup

We train our depth completion model for 40 epochs until convergence using RMSprop as the optimizer with learning rate $\rho := 10^{-4}$ and batch size of 8 (SDS-ST1k) or 4 (NYU-Depth v2). Given its very large resolution, all trainings of our model on SDS-ST1k leverage patches of 160×160, but we switch to full 640×480 inference when

computing quality metrics since our networks are fully convolutional.

We also retrain other competing methods for comparison, by following the supervised training procedures described by the respective authors as reproducible from their codes (S2D, CSPN and NLSPN) or descriptions (D^3), while providing as input our sparse ToF datasets. We also tested the authors' CSPN and NLSPN models pretrained on NYU-Depth v2 with randomly sampled patterns, but the resulting performances were worse than our retraining.

In addition to the $\mathrm{float32}$ model, we consider our quantized MP models $\mathrm{W}_8\ \mathrm{A}_8$, $\mathrm{W}_4\ \mathrm{A}_8$, $\mathrm{W}_4\ \mathrm{A}_4$ (Sec. 7.2.4) and $\mathrm{W}_\star$ (Sec. 7.2.3)[‖]; the subscripts denote the respective average bit widths. For MP QAT, we use the NNabla [106] implementation of [93]. In this case we obtained the best results by training our MP models starting from the pretrained $\mathrm{float32}$ model, with the Adam optimizer for 60 epochs until convergence, starting from learning rate $\rho := 10^{-4}$ with cosine learning rate decay scheduling [103].

7.3.2 Depth Completion Models

We first discuss some quantitative results of our method (in $\mathrm{float32}$ precision) against others in Tab. 7.2. We achieve very good results on both evaluated datasets despite their different sparsity level K: our method ranks second-best on NYU-Depth v2 with a RMSE of $63.7\,mm$ and **best** on SDS-ST1k with a RMSE of $98.7\,mm$. MAE and MNS confirm the RMSE ranking. We also achieve very competitive t_{GPU} by virtue of the simple initialization obtained via NNI and simple encoder-decoder architecture of Fig. 7.1a.

NLSPN provides the most competitive performances: it is quality-wise the **best** approach on NYU-Depth v2 (but the RMSE gap w.r.t. ours is very limited, only $2.3\,mm$) and second-best on SDS-ST1k, as confirmed by all quality metrics as well as visual results (Sec. 7.3.4). However, its t_{GPU} is relatively high even if the model is substantially compact. This is due to spatial propagation, which requires several iterations (18 as recommended in [5]) that linearly increase t_{GPU} even at constant memory footprint. t_{GPU} also notably increases significantly between 304×224 and 640×480 resolutions. Moreover, the initialization cost (*i.e.*, the input encoder-decoder module of NLSPN) is substantially higher than that of our method, which runs on CPU.

As for the other methods, it is possible to see that CSPN has slightly but consistently

[‖]In $\mathrm{W}_\star$ activations are left $\mathrm{float32}$.

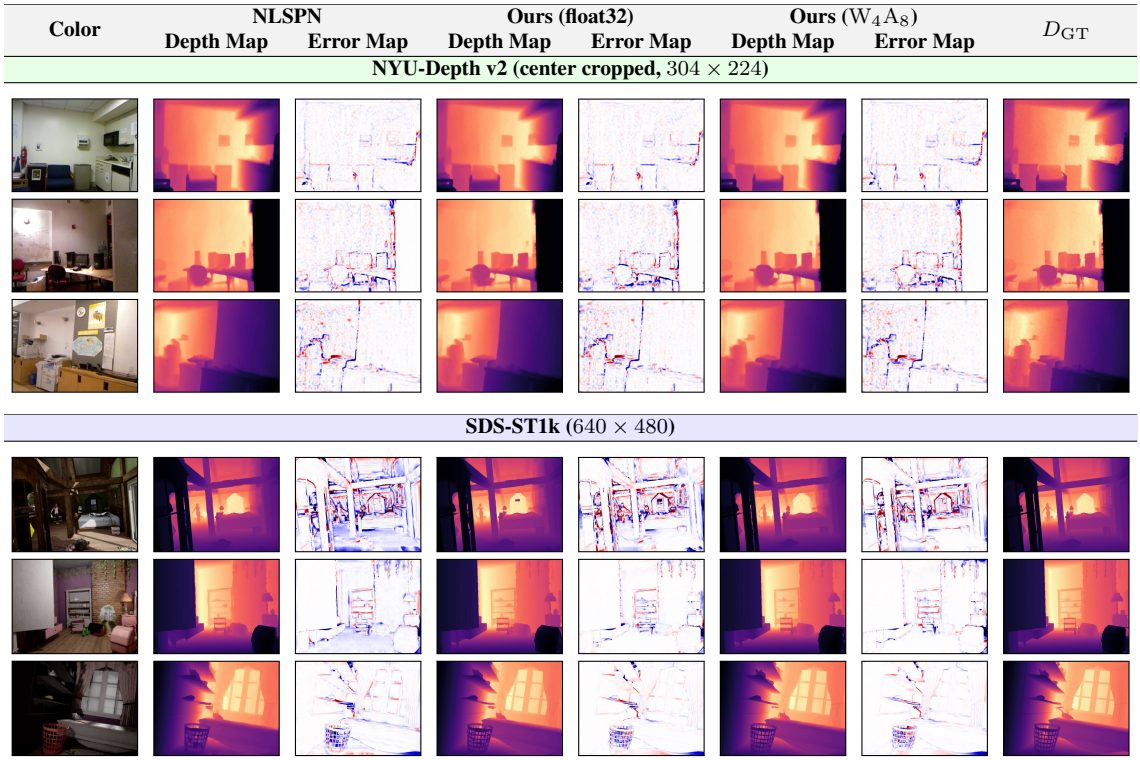


Figure 7.4: Qualitative results. We report, for three arbitrary frames in the test sets of NYU-Depth v2 (top rows) and SDS-ST1k (bottom rows), the predicted depth maps $\hat{D}$ and error maps $\hat{D} - D_{\mathrm{GT}}$ (range: $[-500, 500]mm$). Figure best viewed in color.

inferior performances than NLSPN and ours on both datasets. We also observed that S2D, even if very fast, yields sub-optimal quality metrics on NYU-Depth v2, and poor quality metrics on SDS-ST1k given its larger resolution at lower K. This confirms the findings in [2, Sec. 6.4] that the RMSE of S2D deteriorates for lower sparsity level K and that structured subsampling patterns yield worse performances than random ones. Finally, we observed that D^3, even if starting from the same initialization as ours, yields worse RMSE and MAE.

7.3.3 Quantized Network Models

The QAT results for our MP models are given in Tab. 7.3. Among those models, we highlight our best weights-only MP QAT result for $\mathrm{W}_\star$; the weights size constraint of $14\,\mathrm{MB}$ is met by weights memory size of $13.9\,\mathrm{MB}$ after training. This yields 14-fold memory size reduction w.r.t. the $\mathrm{float32}$ model at $198.5\,\mathrm{MB}$. There, we observe

limited and graceful degradation of the quality metrics[**] (*e.g.*, the RMSE loss is only 1.2 mm on NYU-Depth v2) when reducing precision of the weights and activations in the $\mathrm{float32}$ model. However, as activations are also crucial for reducing memory footprint (Sec. 7.2.4), we remark that the overall most compact model with minimal impact on quality metrics is $\mathrm{W}_4\mathrm{A}_8$ with a RMSE loss of only 0.7 mm on NYU-Depth v2, as $\mathrm{W}_4\mathrm{A}_4$ sees instead significant degradation of the MRE, MNS, and MAE metrics (*e.g.*, 6.6 mm in RMSE). We select this model for the following analysis.

7.3.4 Qualitative Analysis

We now highlight the visual differences observed between different depth completion approaches. We consider the best competitor from Sec. 7.1.2, NLSPN, against our method in its $\mathrm{float32}$ and MP $\mathrm{W}_4\mathrm{A}_8$ flavors. We report three samples from each dataset with both depth maps $\hat{D}$ and error maps $\hat{D} - D_{\mathrm{GT}}$ in the same range to allow for a visual comparison. On NYU-Depth v2 NLSPN yields sharp edges with minimal amount of "mixed depth" pixels. However, the float32 and $\mathrm{W}_4\mathrm{A}_8$ models are as sharp, *e.g.*, on the wall discontinuities (middle row). The error patterns seen in all approaches on far planar regions (top row) are probably due to the acquisition of D_{GT} with a Kinect sensor, whose accuracy decreases with the distance [107].

As for the higher-resolution SDS-ST1k dataset, we see that while depth map quality of our models is comparable to that of NLSPN, on several scenes the latter yields larger errors on planar surfaces (*e.g.*, the ground plane) than ours as shown in the corresponding error maps (top, middle row); our model also recovers more accurately some complex object details such as the cupboard (middle row), trash bin and shelves (bottom row).

7.4 Conclusions

In these two chapters we have described a way of acquiring accurate depth information from iToF cameras at the cost of spatial resolution. We have introduced the task of depth completion, which is specific to these new settings and described a depth completion model based on U-Net architecture. As for MPI correction, we also kept

[**]We do not report t_{GPU} as comparable with that of the $\mathrm{float32}$ model and not indicative of computation time on dedicated hardware, since fixed-precision operations are here emulated by $\mathrm{float32}$ operations as in [94, 93].

its size small, using mixed-precision quantization. Our model performs within a few percent units of the state-of-the-art in terms of quality metrics on standard datasets such as NYU-Depth v2, but achieves faster GPU time at competitively small memory footprint and is suitable for real-time applications. A dedicated hardware implementation of our QNN will take full advantage of mixed-precision by fixed-point operations. We will investigate knowledge distillation with MP QAT to compress the network, as well as domain adaptation techniques [108, 71] to improve depth map quality on real RGB-ToF datasets.

8 Semi-supervised Techniques for Spectrum Reconstruction

In this chapter we will present a work focused on the training of network architectures for spectrum reconstruction using a limited amount of supervision. We will begin with an introduction to hyperspectral imaging, mention some related works on spectrum reconstruction, and then continue by introducing the main task, which is training a deep learning architecture with limited supervision. We will then explain the methods we devised, with the corresponding degree of supervision required and finally compare the performance of our approaches with some of the best performing architectures in the literature.

8.1 Hyperspectral Imaging

The output of a common RGB camera consists of an image with 3 values per pixel, each corresponding to the integration between the spectrum of the incoming light and the corresponding colour matching function. In Figure 8.1, we can see the colour matching functions corresponding to the XYZ colour space. This 3 channel representation is quite common as it mimics the behaviour of the human eye, but it is also a rough encoding of the incoming information. In general, the spectral signature is a complex function, which can hardly be encoded by 3 values and which extends much further than the visible spectrum. An example of it can be seen in Figure 8.2. Hyperspectral cameras provide instead a much finer sampling of the incoming spectrum, which is not necessarily constrained to the visible spectrum. While an RGB camera returns only 3 values for each image pixel, a hyperspectral camera can provide tens or hundreds of samples. The downside of this technology is the added complexity of scanning such a high number of wavelengths which can lead to longer acquisition times.
There are 4 scanning techniques, each with its pros and cons. *Spatial scanning*, employed in push-broom spectrometers [109], gathers simultaneously all the wavelengths

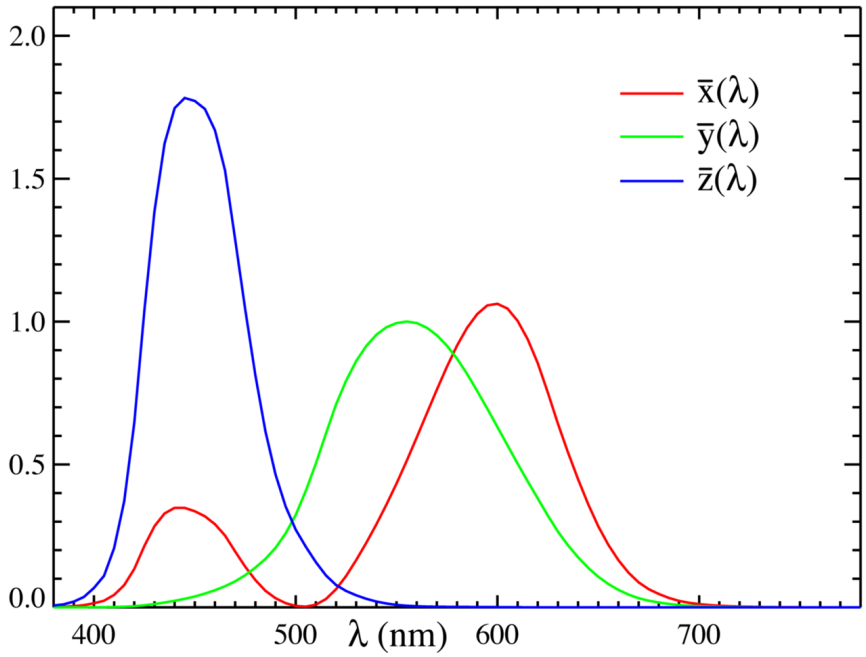


Figure 8.1: Colour matching functions of the XYZ colour space.

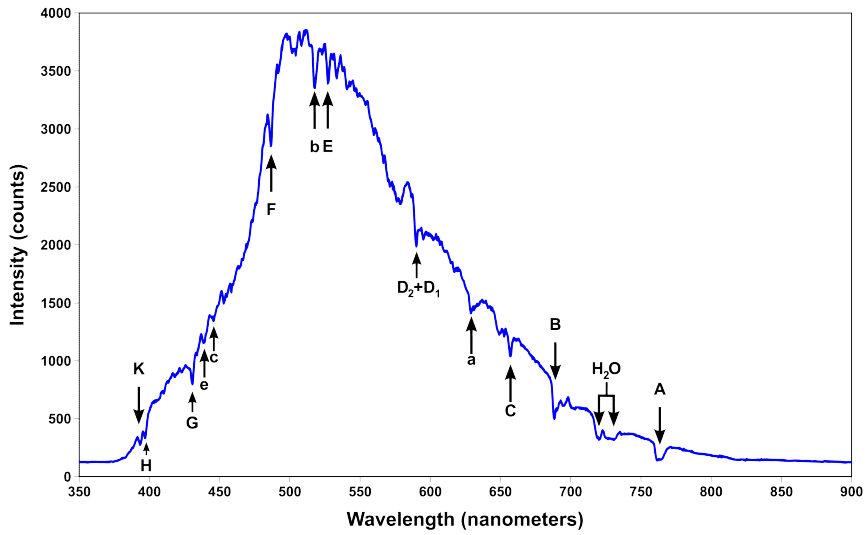


Figure 8.2: Spectral signature of blue sky.

of a single line of the image, while *spectral scanning*, used for example in tunable filters cameras [109], outputs a monochromatic image for each sampled wavelength. These two techniques provide high-resolution images, but have long acquisition times,

as the operation has to be repeated for each image column or for each sampled wavelength respectively. *Non-scanning* is a different kind of technique which takes a snapshot of the entire 3D data-cube, by projecting it into a 2D image; this leads to a fast acquisition time, but requires a pricey setup and long times of post-processing for the recovery of the 3D data cube. Finally, *spatiospectral scanning* is a techniques standing in between spatial and spectral scanning, and it consists in sampling the data-cube in a diagonal manner. An overview of these 4 scanning techniques is shown in Figure 8.3.

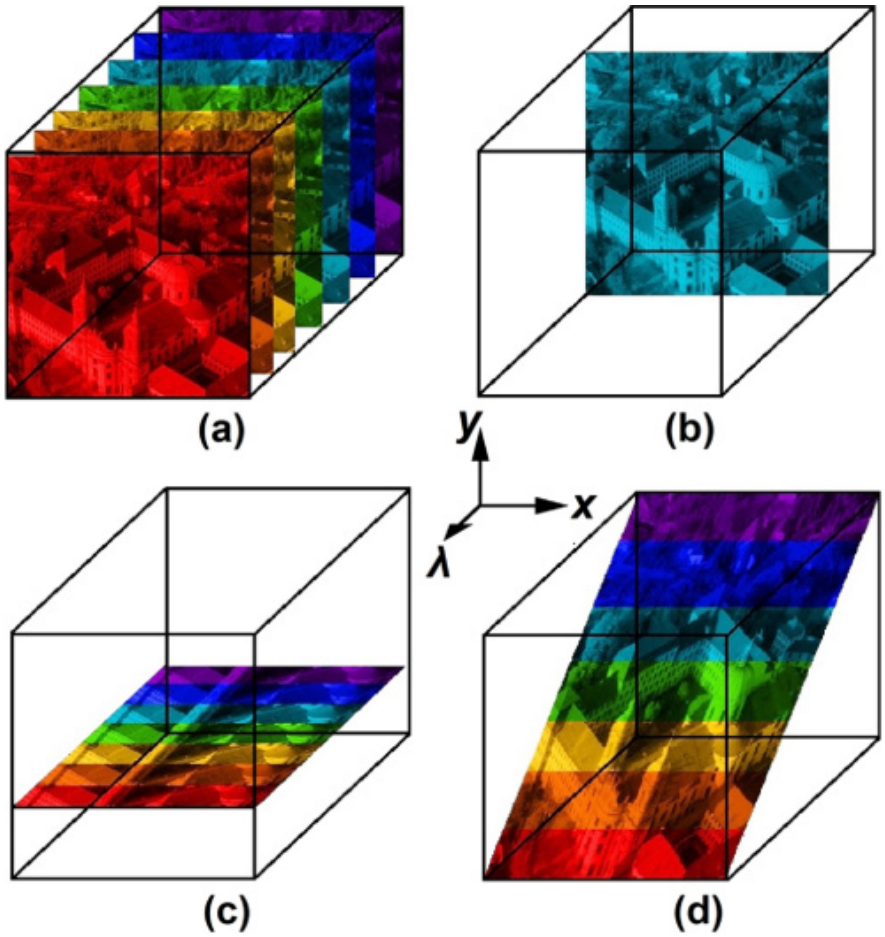


Figure 8.3: Overview of the 4 scanning techniques.(a) *non-scanning*, (b) *spectral scanning*, (c) *spatial scanning*, (d) *spatiospectral scanning*. Image taken from [6].

8.2 Related Work

As we have seen in Section 8.1, all the current scanning techniques for hyperspectral imaging have flaws linked to high cost and/or high acquisition/processing time. To overcome these issues, a lot of focus was put on the processing side to recover the necessary information from low resolution HSI images. In particular hyperspectral super-resolution starts with a low resolution HSI image, and possibly a matching high-resolution RGB one, and tries to enhance it in the spatial domain [110].

On a different note, *spectrum reconstruction* tries to limit as much as possible the use of hyperspectral hardware: at inference time the pipeline would get in input an RGB image and directly provide in output the hyperspectral counterpart [7]. The procedure, shown in Fig. 8.4, can also be seen as image super-resolution in the spectral domain, since it is not the size of the image that changes but the number of channels. Spectrum reconstruction is overall a very ill-posed task since the RGB information (that is the result of an integration operation over the spectrum) is a very sparse representation of the hyperspectral data. One of the first works introducing the recovery of hyperspectral

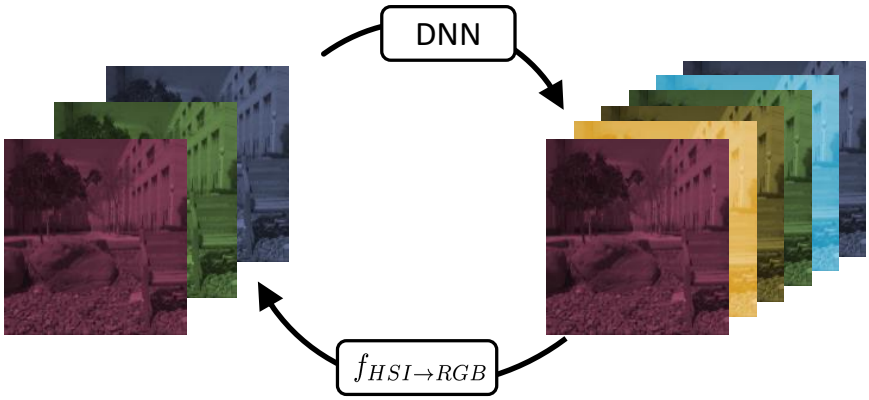


Figure 8.4: Spectrum Reconstruction converts an RGB image to its hyperspectral counterpart. We include the inverse transformation as a physical model within the training.

information from single RGB images following a machine learning approach was [7], where the hyperspectral data is computed using a dictionary based approach. Their contribution also included the acquisition of the ICVL dataset which is one of the biggest up to date. Later, Aeschbacher et al. proposed A+ [111], which built on the previous method and improved its performance: to our knowledge, it has the best

performances among approaches not based on deep learning.

Deep learning techniques were firstly employed for this task in 2017 when Galliani et al. [112] adapted the Tiramisu semantic segmentation network to spectrum reconstruction. Since then, many other architectures have been proposed as in [113, 114, 115] but it was only in 2018 with the NTIRE challenge for spectrum reconstruction [116] that the current state of the art was set: all the participants employed deep learning approaches, adapting architectures such as ResNet [117], U-Net [118] and Generative Adversarial Networks (GANs) [119]. The winner of the challenge was the team of Shi et al. [120], that managed to get first and second place with their HSCNN-D and HSCNN-R architectures based on the residual network model. Note that we use their HSCNN-R architecture as our core network. It is also worth mentioning the U-Net architecture proposed by Can et al. [121] which obtained the best results on the NUS [122] and CAVE [123] datasets outperforming Galliani et al. [112]. Yan et al. [115] also exploit a U-Net architecture and claimed to be on par with [112].

8.3 Training with Limited Supervision

As already pointed out, the biggest issue when trying to exploit machine learning for this task is the lack of hyperspectral data. As a matter of fact, while there exist databases of labeled RGB images for various tasks containing tens of thousand or even millions of samples, the biggest hyperspectral images database is composed of only 256 images [116]. Training a Deep Neural Network with such a limited amount of data is a very challenging task easily leading to overfitting on the training set. A commonly used workaround is to split the training images into patches: this strategy allows to reduce the issue but goes far from solving the problem. Moreover, these large datasets are only available for a few specific cases and it is therefore impossible to deploy such networks in other situations without creating a large HSI dataset which is costly and time-consuming. In this thesis, we introduce a set of learning strategies able to provide good performances even when the HSI data at our disposal is very limited, thus overcoming the issue.

We consider 4 possible scenarios. The first is the completely unsupervised scenario (T_u), where no hyperspectral information is available and we have only RGB data at our disposal. Here, as expected, performances are limited but we will introduce the physical model which will be used as the starting building block for the other

approaches. The second (T_i) is a semi-supervised scenario where a large number of RGB images are at our disposal but only a few hyperspectral ones. In the third setting (T_p) we have the RGB images while the HSI information is given only for a few pixels. Notice that this is a much more challenging setting compared to the previous one since the number of samples with hyperspectral information is several orders of magnitude smaller, but it is interesting since it matches the behaviour of point spectrometers. For the latter we consider a domain transfer setting where HSI information is available for a source domain but only a limited amount of hyperspectral samples are provided for the target domain, that has slightly different properties.

8.4 Proposed Methods

Fig. 8.5 shows a high level overview of our framework. The RGB images are given in input to a Deep Neural Network (DNN) that can be any suitable architecture for spectrum reconstruction. In this work we exploited the HSCNN-R model [120]: this network re-frames the spectral interpolation as a super-resolution task in the spectral domain and has been developed starting from the VDSR network for image super-resolution [124]. The baseline HSCNN network [114] has been improved by removing the hand-crafted upsampling and introducing residual blocks. We further slightly modify the architecture by using a ReLU activation function for the final layer to constraint the output to be positive and a different initialization improving the mapping to the desired $[0, 1]$ range.

When applying machine learning to complex tasks, a good practice is to exploit the underlying physical model to constraint the training [125, 126, 127]. For example when splitting illumination and reflectance in color images it is possible to exploit the fact that the product of the two gives the original image to discard solutions not satisfying this relation [128]. While this kind of constraints are not sufficient on their own, they significantly aid the training by putting some boundaries on the manifold of possible solutions. In order to exploit this idea, the output of the network is connected to multiple branches: the first one performs the conversion to RGB using the physical model and compares the result to the input, the second takes care of the additional constraints driving the reconstruction in the hyperspectral domain and finally there is a supervised loss for cases where HSI data is available.

The physical model branch will remain the same throughout all the proposed tech-

niques, while different strategies will be proposed for the other modules and for the merging between the various components. In the following sections we will firstly introduce the physical model and then present case by case the various proposed strategies.

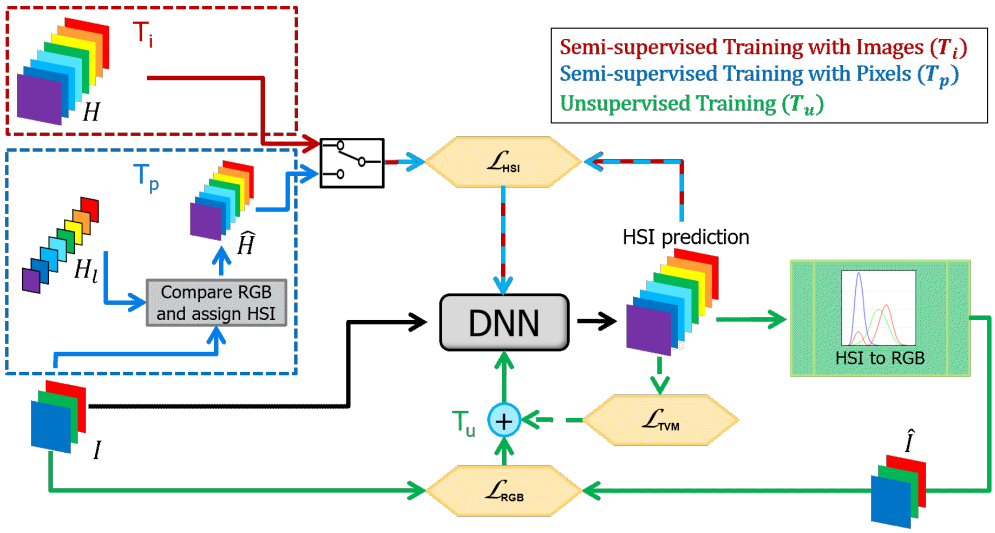


Figure 8.5: Structure of the training pipeline

8.4.1 Physical model

In the case of spectrum reconstruction, the underlying physical model that can be exploited is the conversion of hyperspectral images back to the RGB domain (see Fig. 8.4): while the RGB to hyperspectral mapping is very underconstrained, the opposite relation is an univocal function $f_{HSI \to RGB}$ (more in detail, there is a function $f^c_{HSI \to RGB}$ with $c = R, G, B$ for each of the 3 color components). For this reason we mapped the output of the network back to the RGB space and compared it to the input image. The difference between the two images is captured by the loss $\mathcal{L}_{RGB}$:

$$\mathcal{L}_{RGB} = \frac{1}{3N_iN_j} \sum_{i=1}^{N_i} \sum_{j=1}^{N_j} \sum_{c=1}^{3} |I_{i,j,c} - f^c_{HSI \to RGB}(P_{i,j})| \tag{8.1}$$

where we denote the pixels in the original image with I and the predicted HSI samples with P. N_i and N_j correspond to the size of the image in the spatial dimensions and c loops over the 3 color channels. In a practical case the conversion would need to be done using the camera spectral sensitivity functions, which are usually provided or

known. In our case, since there are no aligned RGB images in the datasets, we employed a standard conversion from the HSI domain to the sRGB domain. We used the CIE 1931 colour matching functions, assumed a D65 standard illuminant and applied gamma correction. However notice that $f_{HSI \to RGB}$ can be any generic function and the proposed approach is agnostic to the employed camera model. As discussed the physical model loss alone is far from sufficient for reliable results and additional constraints are needed. In fact, there is a multitude of spectra which can be converted back to the same RGB values, as the transformation is the output of an integration operation. The physical model does not help in resolving this ambiguity, but penalizes all options not leading to the input RGB values.

8.4.2 Unsupervised Training (T_u)

The training with RGB information alone proved, as expected, to be particularly challenging: our objective is to retrieve the HSI information without having any ground truth spectral sample. Referring to Fig. 8.5, we used the Total Variation Minimization (TVM) loss $\mathcal{L}_{TVM}$ [129] in the spectral dimension as an additional constraint. The TVM loss is usually composed of two terms, one regarding the variation of the new signal and the other keeping into account the similarity with the original signal. We only used the first since the physical model can be seen as a raw replacement of the second:

$$\mathcal{L}_{TVM} = \frac{1}{N_i N_j (N_s - 1)} \sum_{i=1}^{N_i} \sum_{j=1}^{N_j} \sum_{s=1}^{N_s - 1} |P_{i,j,s+1} - P_{i,j,s}| \tag{8.2}$$

where as before N_i and N_j correspond to the two image dimensions while N_s corresponds to the number of spectral bands; the P term indicates the predicted intensity of a pixel for a given position and spectral band.

The network training is performed with the combined loss $\mathcal{L}_u = \mathcal{L}_{RGB} + \lambda \mathcal{L}_{TVM}$. The λ parameter controls the relative weight of the two terms and we experimentally set it to $\lambda = 0.01$. The effect of $\mathcal{L}_{TVM}$ consists in smoothing the signal in the spectral dimension, which is desirable as the spectrum of an image tends to be very correlated and without discontinuities as shown in [130], [131] and [132].

The results are still far from sufficient, but the model trained together with $\mathcal{L}_{RGB}$ and $\mathcal{L}_{TVM}$ presents two useful properties: the output has a consistent RGB representation and shows correlation in the spectral domain (smooth behaviour of the spectra). This

unsupervised model has been used as a starting point for all of our successive tasks, as it significantly eases the training.

8.4.3 Semi-supervised Training with Images (T_i)

When the training dataset contains both RGB and hyperspectral images, the previous approach can be combined with the standard supervised training on hyperspectral data. The training in this case is performed in two phases. In the first only RGB images are used and an initial model is trained for S_1 steps in an unsupervised way as in Section 8.4.2, minimizing the loss $\mathcal{L}_u$. This initial stage performs a robust initialization of the network making the semi-supervised training in the next phase more stable. In the second phase we use together the two different sets of images, the RGB ones and the hyperspectral ones. For images where only RGB data is available, we use the previously introduced physical model loss, but without the total variation term. We disabled this term since while in the initial phase its smoothing effect was needed for a more stable estimate, in the second phase it would lead to an excessive smoothing of the spectra, as a more accurate shape can now be learned from the HSI data. For images where the HSI data is available, we instead use a supervised strategy where we compare the HSI ground truth with the prediction using the Mean Relative Absolute Error (MRAE):

$$\mathcal{L}_{HSI} = \frac{1}{N_i N_j N_s} \sum_{i=1}^{N_i} \sum_{j=1}^{N_j} \sum_{s=1}^{N_s} \frac{|P_{i,j,s} - H_{i,j,s}|}{H_{i,j,s}} \tag{8.3}$$

where N_s is the number of spectral bands (31 in our case), H denotes the ground truth hyperspectral values and P the network prediction. We used the MRAE since this metric normalizes the error w.r.t. the intensity of the ground truth, avoiding a bias towards brighter pixels (as happening with RMSE), especially because it is common for some of the channels to be consistently darker than others.

In particular, each batch of training data (that contains $B_s = 64$ samples in our experimental setting, see Section 8.6.1) contains only images from one set (i.e., with HSI ground truth or RGB only). Assuming to have a much larger amount of RGB images than hyperspectral data, the training procedure is organized in blocks of k iterations: each block consist in a first iteration where the network is trained on a batch of B_s patches with HSI ground truth followed by $k - 1$ iterations where the network

is trained on RGB only data (the RGB batches are different in each iteration). The training procedure is shown in Fig. 8.6. It is repeated for S_2 steps to obtain the final model. Notice how the use of RGB images in training reduces the risk of overfitting: if the network overfits on the HSI images, which are few, still it will have to fulfil the physical model constraint on the much larger RGB dataset.

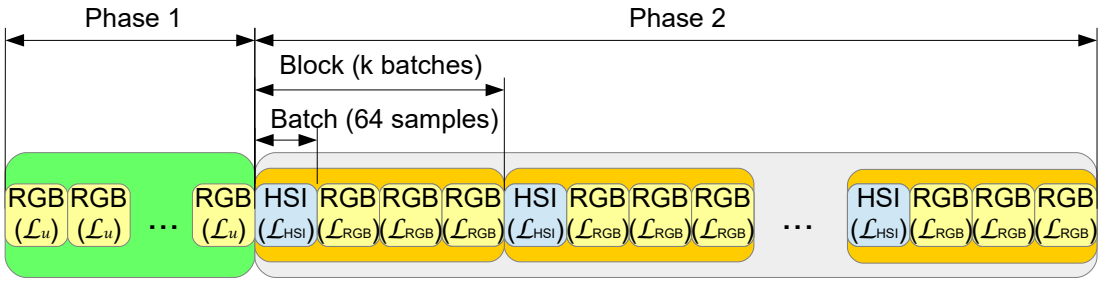


Figure 8.6: Proposed semi-supervised training procedure for the scenarios T_i and T_p (example with $k = 4$).

8.4.4 Semi-supervised Training with Pixels (T_p)

For this more challenging scenario the hyperspectral ground truth is only available for a few pixels scattered among the images, while all the other ones only have RGB information. The major challenge in this case is how to exploit the very little amount of HSI data in a deep learning scenario. This setting reflects the real world situation where the RGB information is easily acquired using standard cameras, while on the hyperspectral side the available hardware is a spectrometer which allows us to acquire only one pixel at a time. Spectrometers are more common and cheaper than devices able to acquire the HSI data of an entire image. In this work we assume to have only N_H hyperspectral pixels. We tested with $10^2, 10^3$ and 10^4 pixels, which are reasonable assumptions for the previously described scenario. We will denote with I_l their RGB values and with H_l their spectra while the selection strategy is discussed in Section 8.5.

The scenario described in Section 8.4.3 for semi-supervision based on images is not suitable for this case. For example the ICVL dataset [7] used in the experimental evaluation contains about 200 images, while in our case we can consider even only $N_H = 10^2$ HSI pixels, i.e., less than a ground truth pixel for each training image. In practice, the information in such a low amount of HSI pixels would be unable to significantly drive the training procedure. The exploited idea consists in building an

approximated ground truth for all of our RGB images using the RGB information together with the small amount of available HSI pixels. In other words, we augment the ground truth available by constructing hyperspectral images based on the few pixels available. For each RGB pixel $I_{i,j}$ we search for the closest one $I_{l(i,j)}$ (in terms of mean absolute error) in the RGB domain among the ones with HSI ground truth. We then assign the HSI information $\hat{H}_{i,j} = H_{l(i,j)}$ found in this way to the original pixel. The procedure is depicted in Fig. 8.7: we are building a coarse approximation of the ground truth where each pixel has the spectrum of the closest sample among the ones with HSI data. At the end of the process each training image will have a roughly estimated HSI counterpart which can be used during training. The training procedure works then in two phases, the first with RGB data only and the second with a mix of RGB images I and estimated HSI data $\hat{H}$. It is the same approach of Section 8.4.3 (Fig. 8.6), using the same losses but with the coarse approximation of the HSI data $\hat{H}$ in place of the ground truth H. In this setting, since HSI data can be estimated for each image, the blocks of iterations are smaller (i.e. $k = 3$), leading to a larger fraction of HSI based iterations. The rounds with RGB information this time are not necessarily beneficial for the final performance but help with the convergence in the cases where the HSI information is very coarse (cases with 1'000 and 100 pixels).

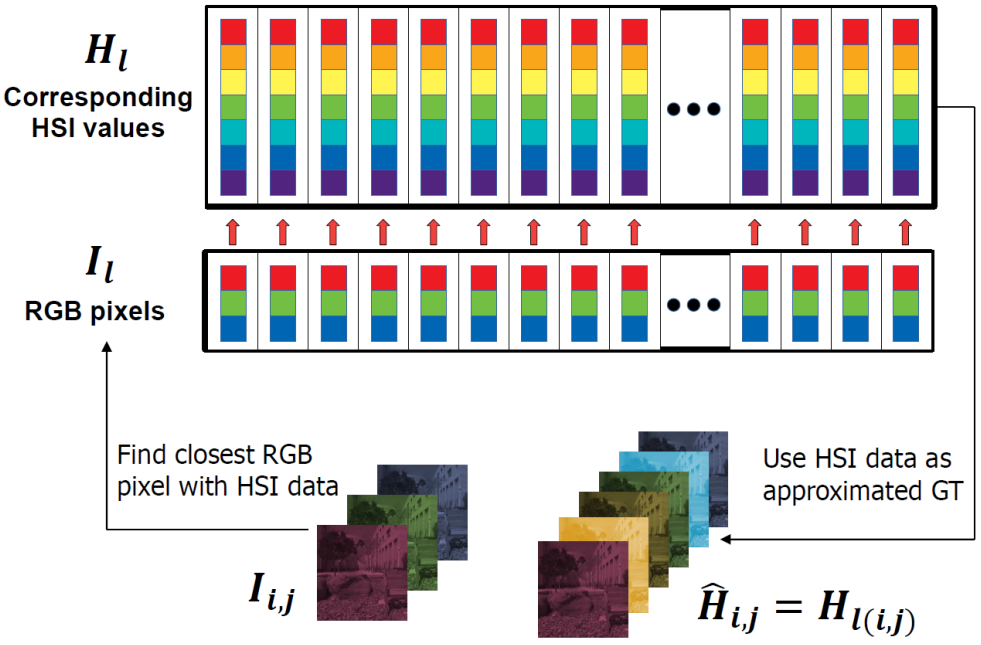


Figure 8.7: Creation of approximate HSI information from a few ground truth pixels.

Notice that this approach makes use of the assumption that pixels with similar RGB values also have a similar spectrum. Even if in some cases the assumption seems reasonable, in particular if they are also part of the same region or object (e.g., pixels from the same region of the sky), this assumption turns out to be false in general, as the same RGB triplet can be originated from very different spectra. However, it provides a reasonably good starting point for the neural network training. For further, more quantitative arguments, refer to the supplementary materials.
Notice that the network does not produce a one-to-one mapping between an input RGB value and the corresponding spectra (otherwise a simple look-up table or clustering technique would lead to the same result), but instead provides an output that depends on the context and semantic interpretation of the neighbourhood of the pixel. In practice we can expect that a small amount of pixels are initially mapped to a completely wrong ground truth; however, later on the network is responsible for detecting and correcting this inaccurate mapping based on the local neighbourhood of the pixels. These expectations are confirmed by the experimental evaluation in Section 8.6.

Moreover, one can argue that the space spanned by the hyperspectral pixels in an image is a considerably small manifold and therefore even if the space of possible hyperspectral pixels is in principle much larger than the RGB space, in practice the sampling of this space is very sparse and it is reasonable to use a limited number of spectra for the supervised loss term using the approximated ground truth. Notice also that we begin our training from a model trained on the $\mathcal{L}_u$ loss and we continue to use the RGB data with the $\mathcal{L}_{RGB}$ loss during the second phase, which guides the network to produce outputs close to the available spectra H' but taking into account the contextual information. Moreover, as remarked in the previous case, also here the RGB constraints helps reducing the risk of overfitting. In short, a simple acquisition with an RGB camera and a few hyperspectral data points with a spectrometer can be used with the proposed approach to train a network efficiently.

8.4.5 Transfer Learning Scenario

Finally the proposed approach has been exploited in a transfer learning framework. We employed the Harvard hyperspectral dataset [132], which is much smaller than the ICVL dataset [7] and not enough for an accurate training of the proposed model. We extracted some hyperspectral pixels as explained in Section 8.5 and these pixels

together with the RGB images were given in input to a network previously trained with full supervision on ICVL data. The training procedure consisted in the same alternating behaviour previously described, based on cycles containing batches using the $\mathcal{L}_{HSI}$ loss on the constructed approximated ground truth and batches using the physical model loss $\mathcal{L}_{RGB}$ on RGB data. The performances for 10^2, 10^3 and 10^4 pixels were compared to a fully supervised training based on a few hyperspectral images and to the proposed pixel loss approach, both performed from scratch.

8.5 Hyperspectral Pixels Selection

The semi-supervised approach with sparse HSI samples (T_p) of Section 8.4.4 requires a reliable strategy for the choice of the pixels with HSI ground truth data.

Recall that we need to select N_H pixels from a set of RGB images which are going to be the pixels with HSI ground truth. The approximated training set will then be built using a nearest neighbour approach, assigning to each RGB pixel the HSI values of the closest point with ground truth. Using euclidean distance in the RGB space as a measure of similarity, if the closest HSI sample is too far away we cannot expect the estimated HSI information to be reliable. Following this line of reasoning a random selection would be suboptimal, as it would lead to some over-represented spectrum types typically corresponding to larger or more frequent regions. As an example, in an outdoor scenario a random choice would provide many very similar HSI pixels from the sky, while probably missing many small structures.

What we are looking for is a way to find a set of pixels which covers the RGB space of our images in the best possible way: ideally we would like the great majority of the pixels of the images to have a close neighbor in the ground truth set. This problem can be solved exploiting a clustering scheme. The RGB pixels are clustered using the k-means algorithm (with the k-means++ [133] initialization) and we compute the centroids as the points minimizing the average within class variance. The spectrum corresponding to the centroid is then assigned to all RGB pixels in the vicinity. The well-known limitations of this simple but fast algorithm are not relevant in our case as we know the number of centroids beforehand and as a spherical shape of the clusters is reasonable.

As an additional remark, the sRGB domain does not seem to be the optimal choice to run the k-means algorithm as the distance in the space does not always match the

perceived color difference. However, a clustering based on the CIELab color space did not improve the results.

Besides using the proposed approach to produce the training sets for the experiments, an important aspect is how to apply this clusterization approach in a practical scenario. One could proceed by acquiring RGB data, cluster them to get the most relevant points and finally get the spectral information for the selected point, typically by using a spectrometer.

Another possibility would be to directly acquire the desired pixels based on the choice of an expert that controls the spectrometer. This would probably lead to even better performances than the ones shown here. since the algorithm is only based on the RGB domain, while an human expert has also some prior knowledge of the semantic content and hyperspectral prior of the elements of the image.

8.6 Experimental Results

In this section we evaluate the performances of our approach on the ICVL dataset [7] and we compare it with the methods of Aeschbacher et al. [111], Can et al. [121], Yan et al. [115] and Shi et al. [120]. The latter is the winner of the NTIRE challenge and the current top performing approach. Notice that, in contrast to ours, these methods are all fully supervised approaches.

8.6.1 Datasets and Experimental Setup

In order to test the proposed approaches we rely on the well-known ICVL dataset [7] which is one the largest hyperspectral dataset up to now along with its extended version used for the NTIRE challenge [116]. It consists of 201 images, mostly outdoor, representing rural and urban scenes. Each HSI image consists of 31 planes representing spectral nodes between 400 and 700 nm by steps of 10 nm. For all experiments both HSI and RGB images were normalized to the $[0, 1]$ range.

The proposed model has been implemented using Pytorch [134]. We exploited the Adam optimization algorithm with an initial learning rate of $2 * 10^{-4}$ and a polynomial function as decay policy. The employed parameters are: $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. The model has been trained in an unsupervised way for $S_1 = 1$'000 steps followed by $S_2 = 2$'000 steps of semi-supervised optimization. Early stopping was

used to reduce overfitting. The training took on average 36 hours using 4 NVIDIA GTX 1080Ti GPUs. The images were split into patches of 50 by 50 pixels and grouped in batches of 64. The inference on the other hand was performed on the entire images. To evaluate the fine-tuning approach described in Section 8.4.5, we used the Harvard dataset [132]. This smaller set is composed of 50 images both outdoor and indoor. For our experiments we only focused on the 39 outdoor images.

For a fair comparison with competing approaches, we tested the different methods on the train and test split used in [121] except for 8 images that we moved from the test set to a validation set, and used the code provided by the authors of the various works with the parameters suggested in the publications. We retrained these methods instead of getting the data from the respective publications since each of them has different variations of the metrics and different splits between training, validation and testing sets. However results are very similar to the ones presented in the papers. For [111] we do not correct the offset in the hyperspectral images and we do not apply the gamma correction, since they apply the hyperspectral to RGB transformation not only to the images but also to the dictionary itself. Therefore, only for this method, the sRGB values are slightly different, but we experimentally verified that this has a very small influence on the results. We compare the methods by using the MRAE metric (8.3) and the Root Mean Squared Error (RMSE).

8.6.2 Quantitative Results

Table 8.1 presents the hyperspectral reconstruction error of the proposed semi-supervised approaches and of the competing methods while Fig. 8.8 gives a visual overview of the performances of the various approaches by comparing the accuracy of each method together with the amount of supervision necessary. Note that the amount of supervision is expressed on a logarithmic scale, as our approaches work with an amount of hyperspectral data several orders of magnitude smaller than competing ones.

The results in the table show that the use of the physical model to include standard RGB images is a very efficient approach to reduce the amount of hyperspectral images or pixels needed for supervision. With only 10 HSI images instead of almost 100 used by the other methods, we obtain a MRAE of 0.019, outperforming all competing methods except [120], that outperforms ours by a very small margin. By using only pixel-wise supervision, allowing the use of data acquired by a spectrometer, our approach

Method	HSI GT	MRAE $*10^2$	RMSE $*10^2$
Aesch. et al. [111]	$3 * 10^6$ pixels	3.3	0.74
Yan et al. [115]	99 images	3.0	0.69
Can et al. [121]	99 images	2.2	0.59
Shi et al. [120]	99 images	**1.4**	**0.48**
	10 images	2.5	0.75
Ours (T_i)	10 images	1.9	0.61
Ours (T_p)	100 pixels	2.5	0.72
	1'000 pixels	1.9	0.66
	10'000 pixels	**1.7**	**0.55**

Table 8.1: Reconstruction error of our semi-supervised methods and of fully-supervised competing methods. The amount of HSI ground truth (GT) used for supervision is also shown.

is very competitive as well. In particular, we get our best result (0.017) using 10'000 pixels only, which is drastically less HSI data than the other supervised approaches. This value is very close to the state-of-the-art and outperforms all approaches except [120]. Furthermore with only 100 pixels we are still able to outperform 2 out of 4 compared methods. The RMSE values give the same key insights as MRAE.

For all benchmarked fully supervised methods, many hyperspectral images are needed for training, thus requiring a hyperspectral camera and a lot of effort for the acquisition. Our approaches are much easier to deploy in a real situation as the hyperspectral data required is hugely reduced. Training the method from Shi et al. on only 10 images leads to a very large drop in accuracy. On the other side, our approach is able to work even with this limited hyperspectral information thanks to the exploitation of standard RGB images with the help of the physical model and reach a much lower error rate (MRAE of 0.019 compared to 0.025). Notice that in this setting, where the amount of HSI data is the same, we clearly outperform [120]. Our pipeline enables the estimation of hyperspectral images in real word situations where the amount of hyperspectral ground truth data is limited. Notice that our method gives the second best result with much less supervision than all the four compared methods. Interesting is the fact that 10'000 pixels selected from 100 images seem to be more representative of the different spectra than 10 full images as it leads to a lower error. If comparing the accuracy with 10'000 pixels selected only from the same 10 images, the image-wise supervision leads to better performances as expected.

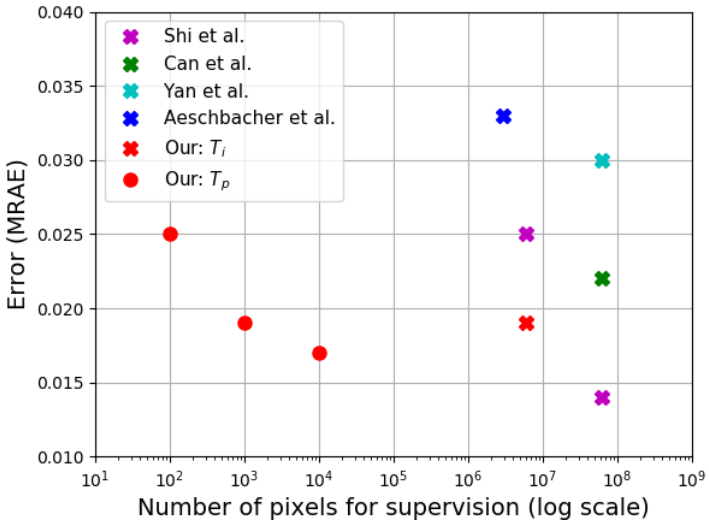


Figure 8.8: MRAE as a function of the amount of supervision necessary to train the DNN or the dictionary for our approaches (in red) and the competing ones.

As a further comparison, using only the clustering approach without the network, e.g., just a lookup table on the spectral values getting the nearest neighbour for each pixel, leads to much worse performances (MRAE of 0.017 instead of 0.029 in the 10'000 pixels case). This shows how the DNN exploits contextual information to recover pixels that were initially assigned to the wrong spectrum.

The results shown in Table 8.1 and Fig. 8.8 are further illustrated in Fig. 8.9, where the approaches using a supervision in the form of images are compared. The number of RGB and hyperspectral pixels used for training are represented as bars, showing that our approach replaces most of the HSI data by the use of standard RGB information. It clearly shows that our approach is making the best of the limited supervision. On one hand our training with only 10 hyperspectral images outperforms all tested approaches from literature expect for [120], all using 99 images. On the other hand it gives much lower error than [120] trained on 10 images (that in this case achieves a MRAE of 0.025), proving that the inclusion of RGB images in the training using the physical model is useful.

The same representation is used in Fig. 8.10 where we compare methods based on pixel-wise supervision along with the state-of-the-art method by Shi et al. [120]. The small amount of hyperspectral data needed by our approach can be in the form of single pixels, while preserving an accurate estimation of hyperspectral data from RGB.

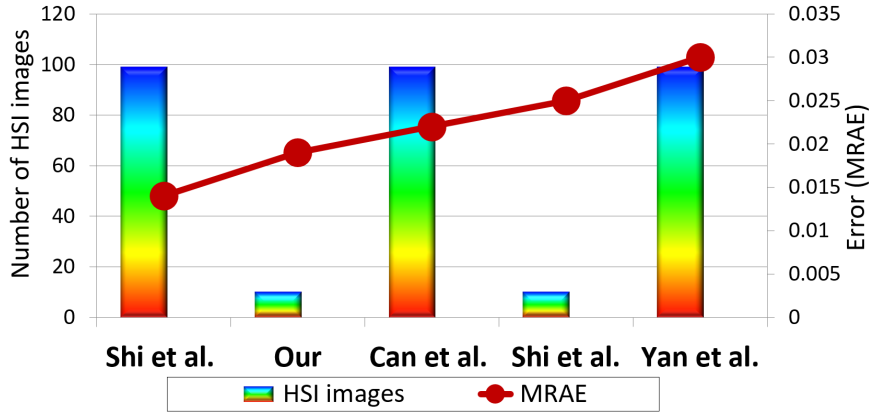


Figure 8.9: MRAE for the fully-supervised competing approaches and for our semi-supervised methods. For each approach the amount of HSI images used for supervision is shown

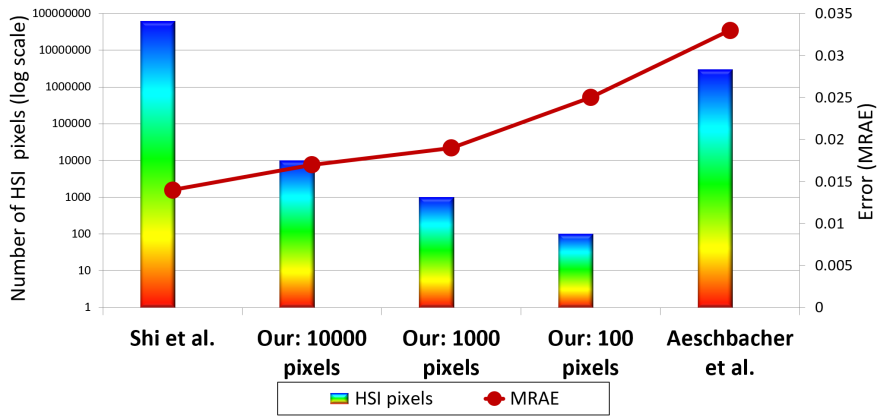


Figure 8.10: Comparison of the fully-supervised external methods and our method semi-supervised with single hyperspectral pixels. The bars correspond to the number of hyperspectral pixels used (log scale) while the points to the MRAE.

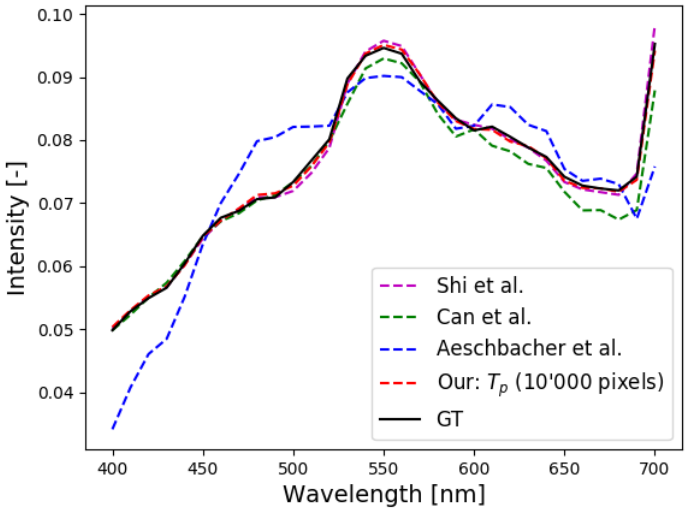


Figure 8.11: Example of spectrum reconstruction for a grass pixel.

As expected the more pixels we have for supervision, the better the results (up to a certain limit). As the number of pixels used grows to 10'000 the error rate decreases and approaches the results of [120], while requiring 4 orders of magnitude less HSI information.

8.6.3 Qualitative Results

Fig. 8.11 shows an example of estimated spectra of a pixel corresponding to the grass region. Consistently with the results discussed before, and similarly to the behaviour on many other samples, the method of [111] is less accurate, followed by the methods of [115] and [121]. Our method gives similar results as [120] although it is only weakly supervised.

Fig. 8.12 shows the average MRAE at each location on some sample images for the different methods. Aeschbacher et al. [111] fails in regions such as grass or sky, where the other methods show good performances. In general the trees are challenging for all the methods. On these images our method shows similar performances as [121] and close to [120].

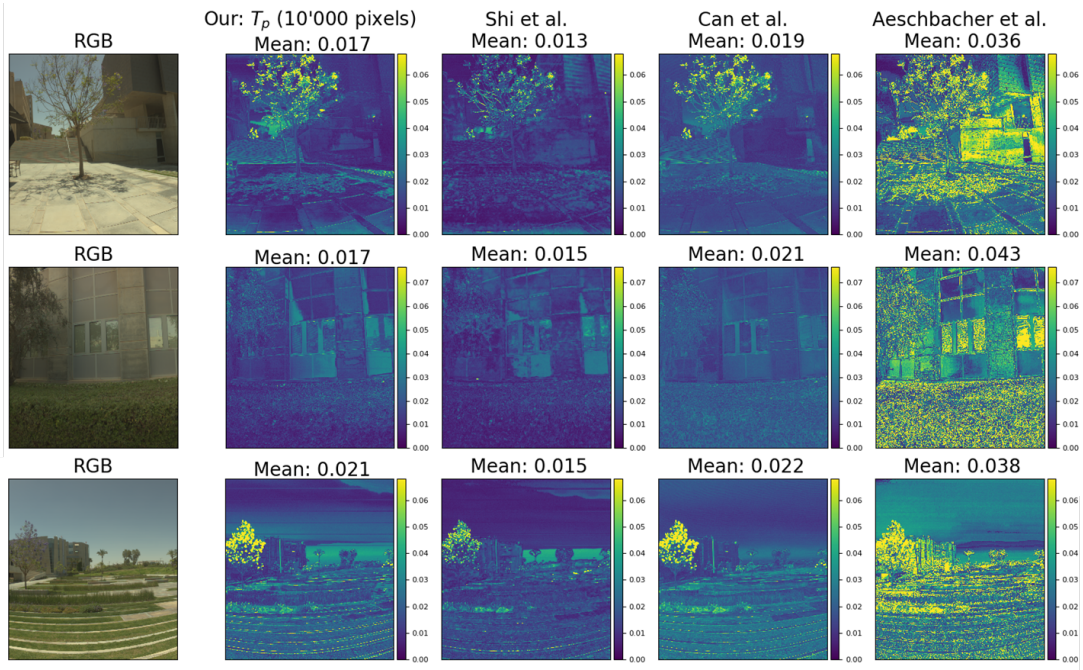


Figure 8.12: Error on 3 sample images from the ICVL dataset [7]. The images show the spatial distribution of the error for our method and for the competing ones.

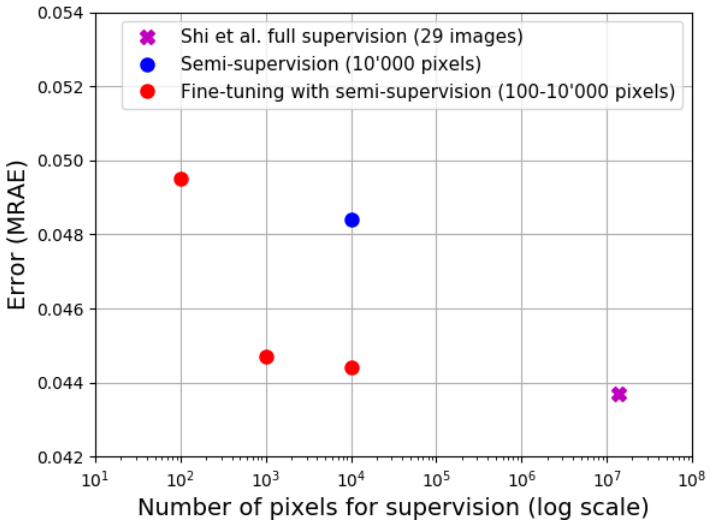


Figure 8.13: Overview of the results on Harvard dataset. The error, as MRAE, is plotted against amount of pixels used for supervision necessary to train the methods (log scale).

8.6.4 Fine-tuning on Harvard dataset

Fig. 8.13 shows an overview of the results obtained by fine-tuning on the Harvard dataset. Note that the amount of supervision is expressed on a logarithmic scale, as our approach works with a few pixels versus the few dozen images needed for the fully supervised case. The results show that our approach can also be used as a domain adaptation tool. It allows to get a reliable prediction when starting from a pre-trained DNN and adapting to a new dataset with only few hyperspectral samples available for it (while standard RGB images are available). The figure shows how the fine-tuning of a model trained initially on ICVL database exploiting only a few pixels of the target domain is almost as reliable as the same model fine-tuned in a fully-supervised fashion (with as little as 10^4 pixels the MRAE is 0.0247 instead of 0.0242). Furthermore the fine-tuning gives slightly better results than a network trained from scratch using the same number of pixels.

8.7 Conclusions and Future Work

In this chapter we introduced a semi-supervised deep learning approach for estimating hyperspectral data from RGB images that can be trained with a limited amount of supervision. We exploited a physical model to constrain the training thus allowing the usage of standard RGB images to support the training procedure. The proposed method allows to obtain very good performances with just a few hyperspectral images or even just some sparse samples. In a real world scenario this allows to obtain a reliable hyperspectral estimation for the whole image using training data from a standard RGB camera coupled with a point spectrometer.

Future research will be devoted to the inclusion of adversarial learning techniques into the proposed framework and to the extension of the fine-tuning approach to a more advanced semi-supervised domain adaptation scheme. Also we plan to test the proposed approach in different environments, for example indoor or including specific objects. Finally, the proposed approach can be extended to make predictions out of the visible range, for example in the near infrared.

9 Conclusions

This Ph.D. thesis is the result of three years of work on Time-of-Flight sensors. We have begun by describing in Chapter 2 the fundamental principles of this technology both in the case of direct ToF sensors and in the case of indirect ones. We then provided a more in depth view of iToF sensors, explaining their limitations and error sources, with the spotlight taken by Multi-Path Interference. Following that, in chapter 3 we introduced the task of MPI correction and that of transient reconstruction and in chapter 4 the datasets we employed; in particular, we also described the *Walls* dataset, a transient dataset based on simple structure that was built during the first year of Ph.D.. In Chapter 5 we then introduced two models for MPI correction. The first one called the *Two-Peaks Network*, which exploits a simple encoding of transient information to aid the training; the second one instead, called *Direct-Global Separation Network*, subdivides the transient information in the iToF domain and also provides a prediction of the entire transient vector. Both approaches were shown to have very competitive performance, with the second one reaching state-of-the-art results on real data.
In Chapter 6, we explained a second manner to employ iToF sensors in order to get a more accurate prediction at the cost of spatial resolution. We then introduced the task of depth completion and showed in Chapter 7 a quantized neural network with competitive performance. For all these methods we posed particular attention to the network size, as a small size is needed to limit the energy consumption and to go towards real time implementations. Finally, in Chapter 8, we investigated a different kind of sensor: hyperspectral cameras. We then showed some techniques for training network for spectrum reconstruction with little supervision.
This thesis leaves open several interesting paths of research. First of all, one straightforward improvement on the *Direct-Global Separation Network*, would be to use an unsupervised domain adaptation approach as done in [39] to get a better performance on real data. Regarding transient reconstruction instead, it is possible to think of em-

ploying a different architecture rather than a parametric model to improve the accuracy of the global reconstruction; afterwards, it would be interesting to see how the global information can be used for other tasks such as material estimation or Non-Line-of-Sight-Imaging. On the topic of depth completion it would be interesting to test the architecture on an actual mobile environment, see if the inference time and the energy consumption are satisfying or if there is still more to improve. Finally, for the semi-supervised techniques on spectrum reconstruction, it would be reasonable to apply UDA approaches to improve the reconstruction.

Bibliography

[1] F. Gutierrez-Barragan, H. G. Chen, M. Gupta, A. Velten, and J. Gu, "itof2dtof: A robust and flexible representation for data-driven time-of-flight imaging," *CoRR*, vol. abs/2103.07087, 2021. [Online]. Available: https://arxiv.org/abs/2103.07087

[2] F. Ma, G. V. Cavalheiro, and S. Karaman, "Self-Supervised Sparse-to-Dense: Self-Supervised Depth Completion from LiDAR and Monocular Camera," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 3288–3295.

[3] Z. Chen, V. Badrinarayanan, G. Drozdov, and A. Rabinovich, "Estimating depth from rgb and sparse sensing," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2018, pp. 167–182.

[4] X. Cheng, P. Wang, and R. Yang, "Learning depth with convolutional spatial propagation network," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 10, pp. 2361–2379, 2019.

[5] J. Park, K. Joo, Z. Hu, C.-K. Liu, and I. So Kweon, "Non-local spatial propagation network for depth completion," in *Proceedings of European Conference on Computer Vision (ECCV)*. Springer, 2020, pp. 120–136.

[6] S. Grusche, "Basic slit spectroscope reveals three-dimensional scenes through diagonal slices of hyperspectral cubes," *Appl. Opt.*, vol. 53, no. 20, pp. 4594–4603, Jul 2014. [Online]. Available: http://opg.optica.org/ao/abstract.cfm?URI=ao-53-20-4594

[7] B. Arad and O. Ben-Shahar, "Sparse recovery of hyperspectral signal from natural rgb images," in *Proceedings of European Conference on Computer Vision*

(ECCV), B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 19–34.

[8] Q. Zhu, L. Chen, Q. Li, M. Li, A. Nchter, and J. Wang, "3d lidar point cloud based intersection recognition for autonomous driving," in *2012 IEEE Intelligent Vehicles Symposium*, 2012, pp. 456–461.

[9] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[10] E. Bostanci, N. Kanwal, and A. F. Clark, "Augmented reality applications for cultural heritage using kinect," *Human-centric Computing and Information Sciences*, vol. 5, no. 1, pp. 1–18, 2015.

[11] Y. M. Kim, C. Theobalt, J. Diebel, J. Kosecka, B. Miscusik, and S. Thrun, "Multi-view image and tof sensor fusion for dense 3d reconstruction," in *Proceedings of International Conference on Computer Vision Workshops (ICCVW)*, 2009, pp. 1542–1549.

[12] C. Kerl, M. Souiai, J. Sturm, and D. Cremers, "Towards illumination-invariant 3d reconstruction using tof rgb-d cameras," in *2014 2nd International Conference on 3D Vision*, vol. 1, 2014, pp. 39–46.

[13] F. Amzajerdian, D. Pierrottet, L. Petway, G. Hines, and V. Roback, "Lidar systems for precision navigation and safe landing on planetary bodies," in *International Symposium on Photoelectronic Detection and Imaging 2011: Laser Sensing and Imaging; and Biological and Medical Applications of Photonics Sensing and Imaging*, F. Amzajerdian, W. Chen, C. Gao, and T. Xie, Eds., vol. 8192, International Society for Optics and Photonics. SPIE, 2011, pp. 27 – 33. [Online]. Available: https://doi.org/10.1117/12.904062

[14] U. R. Dhond and J. K. Aggarwal, "Structure from stereo-a review," *IEEE transactions on systems, man, and cybernetics*, vol. 19, no. 6, pp. 1489–1510, 1989.

[15] R. Horaud, M. Hansard, G. Evangelidis, and M. Clment, "An overview of depth cameras and range scanners based on time-of-flight technologies," *Machine Vision and Applications*, vol. 27, 10 2016.

[16] R. O. Dubayah and J. B. Drake, "Lidar remote sensing for forestry," *Journal of Forestry*, vol. 98, no. 6, pp. 44–46, 2000.

[17] P. Zanuttigh, G. Marin, C. Dal Mutto, F. Dominio, L. Minto, and G. M. Cortelazzo, "Time-of-flight and structured light depth cameras," *Technology and Applications*, pp. 978–3, 2016.

[18] https://www.sony.de/electronics/smartphones/xperia-1m2.

[19] D. Freedman, Y. Smolin, E. Krupka, I. Leichter, and M. Schmidt, "Sra: Fast removal of general multipath for tof sensors," in *Proceedings of European Conference on Computer Vision (ECCV)*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 234–249.

[20] S. Su, F. Heide, G. Wetzstein, and W. Heidrich, "Deep end-to-end time-of-flight imaging," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[21] G. Agresti and P. Zanuttigh, "Deep learning for multi-path error removal in tof sensors," in *Proceedings of European Conference on Computer Vision Workshops (ECCVW)*, September 2018.

[22] E. Buratto, A. Simonetto, G. Agresti, H. Schfer, and P. Zanuttigh, "Deep learning for transient image reconstruction from tof data," *Sensors*, vol. 21, no. 6, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/6/1962

[23] A. Simonetto, G. Agresti, P. Zanuttigh, and H. Schäfer, "Lightweight deep learning architecture for mpi correction and transient reconstruction," *IEEE Transactions on Computational Imaging*, 2022.

[24] X. Jiang, V. Cambareri, G. Agresti, C. I. Ugwu, A. Simonetto, F. Cardinaux, and P. Zanuttigh, "A low memory footprint quantized neural network for depth completion of very sparse time-of-flight depth maps," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 2687–2696.

[25] A. Simonetto, P. Zanuttigh, V. Parret, P. Sartor, and A. Gatto, "Semi-supervised deep learning techniques for spectrum reconstruction," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 7767–7774.

[26] F. Blais, "Review of 20 years of range sensor development," *Journal of Electronic Imaging*, vol. 13, no. 1, pp. 231 – 243, 2004. [Online]. Available: https://doi.org/10.1117/1.1631921

[27] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[28] B. F. Aull, A. H. Loomis, D. J. Young, R. M. Heinrichs, B. J. Felton, P. J. Daniels, and D. J. Landers, "Geiger-mode avalanche photodiodes for three-dimensional imaging," *Lincoln laboratory journal*, vol. 13, no. 2, pp. 335–349, 2002.

[29] C. L. Niclass, A. Rochas, P.-A. Besse, and E. Charbon, "A cmos single photon avalanche diode array for 3d imaging," in *2004 IEEE International Solid-State Circuits Conference (IEEE Cat. No. 04CH37519)*. IEEE, 2004, pp. 120–517.

[30] F. Zappa, S. Tisa, A. Tosi, and S. Cova, "Principles and features of single-photon avalanche diode arrays," *Sensors and Actuators A: Physical*, vol. 140, no. 1, pp. 103–112, 2007.

[31] F. Gramuglia, M.-L. Wu, C. Bruschini, M.-J. Lee, and E. Charbon, "A low-noise cmos spad pixel with 12.1 ps sptr and 3 ns dead time," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 28, no. 2: Optical Detectors, pp. 1–9, 2022.

[32] "Canon successfully develops the worlds first 1-megapixel spad sensor," https://global.canon/en/technology/spad-sensor-2021.html.

[33] F. Piron, D. Morrison, M. R. Yuce, and J.-M. Redouté, "A review of single-photon avalanche diode time-of-flight imaging sensor arrays," *IEEE Sensors Journal*, vol. 21, no. 11, pp. 12 654–12 666, 2020.

[34] F. Ceccarelli, G. Acconcia, A. Gulinatti, M. Ghioni, I. Rech, and R. Osellame, "Recent advances and future perspectives of single-photon avalanche diodes for quantum photonics applications," *Advanced Quantum Technologies*, vol. 4, no. 2, p. 2000102, 2021.

[35] M. Gupta, S. K. Nayar, M. B. Hullin, and J. Martin, "Phasor imaging: A generalization of correlation-based time-of-flight imaging," *ACM Trans. Graph.*, vol. 34, no. 5, 2015. [Online]. Available: https://doi.org/10.1145/2735702

[36] A. P. P. Jongenelen, D. A. Carnegie, A. D. Payne, and A. A. Dorrington, "Maximizing precision over extended unambiguous range for tof range imaging systems," in *2010 IEEE Instrumentation Measurement Technology Conference Proceedings*, 2010, pp. 1575–1580.

[37] Kilho Son, M. Liu, and Y. Taguchi, "Learning to remove multipath distortions in time-of-flight range images for a robotic arm setup," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 3390–3397.

[38] J. Marco, Q. Hernandez, A. Muñoz, Y. Dong, A. Jarabo, M. H. Kim, X. Tong, and D. Gutierrez, "Deeptof: Off-the-shelf real-time correction of multipath interference in time-of-flight imaging," *ACM Trans. Graph.*, vol. 36, no. 6, 2017. [Online]. Available: https://doi.org/10.1145/3130800.3130884

[39] G. Agresti, H. Schaefer, P. Sartor, and P. Zanuttigh, "Unsupervised domain adaptation for tof data denoising with adversarial learning," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[40] M. Lindner, I. Schiller, A. Kolb, and R. Koch, "Time-of-flight sensor calibration for accurate range sensing," *Computer Vision and Image Understanding*, vol. 114, no. 12, pp. 1318–1328, 2010.

[41] S. Fuchs, "Multipath interference compensation in time-of-flight camera images," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 3583–3586.

[42] S. Fuchs, M. Suppa, and O. Hellwich, "Compensation for multipath in tof camera measurements supported by photometric calibration and environment integration," in *Computer Vision Systems*, M. Chen, B. Leibe, and B. Neumann, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 31–41.

[43] D. Jiménez, D. Pizarro, M. Mazo, and S. Palazuelos, "Modeling and correction of multipath interference in time of flight cameras," *Image and Vision Computing*, vol. 32, no. 1, pp. 1–13, 2014.

[44] A. Bhandari, M. Feigin, S. Izadi, C. Rhemann, M. Schmidt, and R. Raskar, "Resolving multipath interference in kinect: An inverse problem approach," in *SENSORS, 2014 IEEE*, 2014, pp. 614–617.

[45] Q. Guo, I. Frosio, O. Gallo, T. Zickler, and J. Kautz, "Tackling 3d tof artifacts through learning and the flat dataset," in *Proceedings of European Conference on Computer Vision (ECCV)*, September 2018.

[46] G. Dong, Y. Zhang, and Z. Xiong, "Spatial hierarchy aware residual pyramid network for time-of-flight depth denoising," in *Proceedings of European Conference on Computer Vision (ECCV)*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 35–50.

[47] G. Agresti, H. Schafer, P. Sartor, Y. Incesu, and P. Zanuttigh, "Unsupervised domain adaptation of deep networks for tof depth refinement," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 01, pp. 1–1, oct 5555.

[48] F. Heide, L. Xiao, W. Heidrich, and M. B. Hullin, "Diffuse mirrors: 3d reconstruction from diffuse indirect illumination using inexpensive time-of-flight sensors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[49] J. Lin, Y. Liu, M. B. Hullin, and Q. Dai, "Fourier analysis on transient imaging with a multifrequency time-of-flight camera," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[50] Y. Liang, M. Chen, Z. Huang, D. Gutierrez, A. Muñoz, and J. Marco, "A data-driven compression method for transient rendering," in *ACM SIGGRAPH 2019 Posters*, ser. SIGGRAPH '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3306214.3338582

[51] M. Galindo, J. Marco, M. O'Toole, G. Wetzstein, D. Gutierrez, and A. Jarabo, "A dataset for benchmarking time-resolved non-line-of-sight imaging," 2019. [Online]. Available: https://graphics.unizar.es/nlos

[52] M. Pharr, W. Jakob, and G. Humphreys, *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016.

[53] P. I. Wójcik and M. Kurdziel, "Training neural networks on high-dimensional data using random projection," *Pattern Analysis and Applications*, vol. 22, no. 3, pp. 1221–1231, 2019.

[54] B. Liu, Y. Wei, Y. Zhang, and Q. Yang, "Deep neural networks for high dimension, low sample size data," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017, pp. 2287–2293. [Online]. Available: https://doi.org/10.24963/ijcai.2017/318

[55] D. Freedman, Y. Smolin, E. Krupka, I. Leichter, and M. Schmidt, "Sra: Fast removal of general multipath for tof sensors," in *Proceedings of European Conference on Computer Vision (ECCV)*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 234–249.

[56] A. A. Dorrington, J. P. Godbaz, M. J. Cree, A. D. Payne, and L. V. Streeter, "Separating true range measurements from multi-path and scattering interference in commercial range cameras," in *Three-Dimensional Imaging, Interaction, and Measurement*, vol. 7864. International Society for Optics and Photonics, 2011, p. 786404.

[57] Y. Liang, M. Chen, Z. Huang, D. Gutierrez, A. Muñoz, and J. Marco, "A data-driven compression method for transient rendering," in *ACM SIGGRAPH 2019 Posters*, ser. SIGGRAPH '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3306214.3338582

[58] W. Weibull *et al.*, "A statistical distribution function of wide applicability," *Journal of applied mechanics*, vol. 18, no. 3, pp. 293–297, 1951.

[59] S. Cohen and L. Guibas, "The earth mover's distance: Lower bounds and invariance under translation," Stanford University CA Dept. of Computer Science, Tech. Rep., 1997.

[60] S. Xin, S. Nousias, K. N. Kutulakos, A. C. Sankaranarayanan, S. G. Narasimhan, and I. Gkioulekas, "A theory of fermat paths for non-line-of-sight shape reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6800–6809.

[61] J. Diebel and S. Thrun, "An Application of Markov Random Fields to Range Sensing," in *Proc. Advances Neural Inf. Process. Syst. (NeurIPS)*, Y. Weiss, B. Schlkopf, and J. Platt, Eds., vol. 18. MIT Press, 2005. [Online]. Available: https://proceedings.neurips.cc/paper/2005/file/353de26971b93af88da102641069b440-Paper.pdf

[62] J. T. Barron and B. Poole, "The Fast Bilateral Solver," in *Proceedings of European Conference on Computer Vision (ECCV)*. Springer International Publishing, 2016, pp. 617–632.

[63] F. Ma and S. Karaman, "Sparse-to-Dense: Depth Prediction from Sparse Depth Samples and a Single Image," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 4796–4803.

[64] M. Jaritz, R. Charette, E. Wirbel, X. Perrotton, and F. Nashashibi, "Sparse and Dense Data with CNNs: Depth Completion and Semantic Segmentation," in *Proc. IEEE Int. Conf. 3D Vis. (3DV)*, 2018, pp. 52–60.

[65] J. Qiu, Z. Cui, Y. Zhang, X. Zhang, S. Liu, B. Zeng, and M. Pollefeys, "DeepLiDAR: Deep Surface Normal Guided Depth Prediction for Outdoor Scene From Sparse LiDAR Data and Single Color Image," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[66] Y. Yang, A. Wong, and S. Soatto, "Dense Depth Posterior (DDP) From Single Image and Sparse Range," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[67] S. Imran, Y. Long, X. Liu, and D. Morris, "Depth Coefficients for Depth Completion," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[68] A. Eldesokey, M. Felsberg, and F. Khan, "Confidence Propagation through CNNs for Guided Sparse Depth Regression," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 10, pp. 2423–2436, Oct. 2020, place: Los Alamitos, CA, USA Publisher: IEEE Computer Society.

[69] A. Eldesokey, M. Felsberg, K. Holmquist, and M. Persson, "Uncertainty-Aware CNNs for Depth Completion: Uncertainty from Beginning to End," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[70] A. Li, Z. Yuan, Y. Ling, W. Chi, S. Zhang, and C. Zhang, "A Multi-Scale Guided Cascade Hourglass Network for Depth Completion," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, 2020, pp. 32–40.

[71] A. Lopez-Rodriguez, B. Busam, and K. Mikolajczyk, "Project to Adapt: Domain Adaptation for Depth Completion from Noisy and Sparse Sensor Data," in *Proceedings of Asian Conference on Computer Vision (ACCV)*, 2020.

[72] C. Qu, T. Nguyen, and C. Taylor, "Depth completion via deep basis fitting," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, 2020, pp. 71–80.

[73] L. Teixeira, M. R. Oswald, M. Pollefeys, and M. Chli, "Aerial Single-View Depth Completion with Image-Guided Uncertainty Estimation," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1055–1062, 2020.

[74] M. Hu, S. Wang, B. Li, S. Ning, L. Fan, and X. Gong, "PENet: Towards Precise and Efficient Image Guided Depth Completion," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 13 656–13 662.

[75] Y. Xu, X. Zhu, J. Shi, G. Zhang, H. Bao, and H. Li, "Depth Completion From Sparse LiDAR Data With Depth-Normal Constraints," in *Proceedings of International Conference on Computer Vision (ICCV)*, 2019.

[76] W. Van Gansbeke, D. Neven, B. De Brabandere, and L. Van Gool, "Sparse and Noisy LiDAR Completion with RGB Guidance and Uncertainty," in *2019 16th International Conference on Machine Vision Applications (MVA)*. IEEE, 2019, pp. 1–6.

[77] S. S. Shivakumar, T. Nguyen, I. D. Miller, S. W. Chen, V. Kumar, and C. J. Taylor, "DFuseNet: Deep Fusion of RGB and Sparse Depth Information for Image Guided Dense Depth Completion," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019.

[78] V. Guizilini, R. Ambrus, W. Burgard, and A. Gaidon, "Sparse auxiliary networks for unified monocular depth prediction and completion," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 11 078–11 088.

[79] A. Wong, X. Fei, S. Tsuei, and S. Soatto, "Unsupervised Depth Completion From Visual Inertial Odometry," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1899–1906, 2020.

[80] V. Patil, W. Van Gansbeke, D. Dai, and L. Van Gool, "Dont Forget The Past: Recurrent Depth Estimation from Monocular Video," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 6813–6820, 2020.

[81] F. Wimbauer, N. Yang, L. von Stumberg, N. Zeller, and D. Cremers, "MonoRec: Semi-Supervised Dense Reconstruction in Dynamic Environments from a Single Moving Camera," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 6108–6118.

[82] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization," in *Proceedings of International Conference on Computer Vision (ICCV)*, 2015, pp. 2938–2946.

[83] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[84] Y. Lin, T. Cheng, Q. Zhong, W. Zhou, and H. Yang, "Dynamic Spatial Propagation Network for Depth Completion," *arXiv preprint arXiv:2202.09769*, 2022.

[85] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016.

[86] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," *arXiv preprint arXiv:1605.04711*, 2016.

[87] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental Network Quantization: Towards Lossless CNNs with Low-Precision Weights," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017.

[88] Z.-G. Liu and M. Mattina, "Learning Low-precision Neural Networks without Straight-Through Estimator (STE)," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2019, pp. 3066–3072.

[89] F. Cardinaux, S. Uhlich, K. Yoshiyama, J. A. Garca, L. Mauch, S. Tiedemann, T. Kemp, and A. Nakamura, "Iteratively Training Look-Up Tables for Network Quantization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 4, pp. 860–870, 2020.

[90] Y. Bai, Y.-X. Wang, and E. Liberty, "ProxQuant: Quantized Neural Networks via Proximal Operators," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019.

[91] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "PACT: Parameterized clipping activation for quantized neural networks," *arXiv preprint arXiv:1805.06085*, 2018.

[92] S. Jain, A. Gural, M. Wu, and C. Dick, "Trained Quantization Thresholds for Accurate and Efficient Fixed-Point Inference of Deep Neural Networks," in *Proceedings of Machine Learning and Systems*, I. Dhillon, D. Papailiopoulos, and V. Sze, Eds., vol. 2, 2020, pp. 112–128. [Online]. Available: https://proceedings.mlsys.org/paper/2020/file/e2c420d928d4bf8ce0ff2ec19b371514-Paper.pdf

[93] S. Uhlich, L. Mauch, F. Cardinaux, K. Yoshiyama, J. A. Garcia, S. Tiedemann, T. Kemp, and A. Nakamura, "Mixed Precision DNNs: All you need is a good parametrization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020. [Online]. Available: https://openreview.net/forum?id=Hyx0slrFvH

[94] M. Nikolić, G. B. Hacene, C. Bannon, A. D. Lascorz, M. Courbariaux, Y. Bengio, V. Gripon, and A. Moshovos, "Bitpruning: Learning bitlengths for aggressive and accurate quantization," *arXiv preprint arXiv:2002.03090*, 2020.

[95] J. Ku, A. Harakeh, and S. L. Waslander, "In Defense of Classical Image Processing: Fast Depth Completion on the CPU," in *2018 15th Conference on Computer and Robot Vision (CRV)*. IEEE, 2018, pp. 16–22.

[96] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor Segmentation and Support Inference from RGBD Images," in *Proceedings of European Conference on Computer Vision (ECCV)*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 746–760.

[97] G. Luetzenburg, A. Kroon, and A. A. Bjørk, "Evaluation of the Apple iPhone 12 Pro LiDAR for an Application in Geosciences," *Scientific Reports*, vol. 11, no. 1, Nov. 2021. [Online]. Available: https://doi.org/10.1038/s41598-021-01763-9

[98] V. Cambareri, "SDS-ST1K Dataset," Sony Depthsensing Solutions NV, Tech. Rep., May 2021. [Online]. Available: https://github.com/sony/ai-research-code

[99] Epic Games, "Unreal Engine (v4.25)." [Online]. Available: https://www.unrealengine.com

[100] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance Transforms of Sampled Functions," *Theory of Computing*, vol. 8, no. 19, pp. 415–428, 2012. [Online]. Available: http://www.theoryofcomputing.org/articles/v008a019

[101] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.

[102] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[103] I. Loshchilov and F. Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017.

[104] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015.

[105] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*. Academic Press, 2014.

[106] A. Hayakawa, M. Ishii, Y. Kobayashi, A. Nakamura, T. Narihira, Y. Obuchi, A. Shin, T. Yashima, and K. Yoshiyama, "Neural Network Libraries: A Deep Learning Framework Designed from Engineers' Perspectives," 2021.

[107] P. Zanuttigh, G. Marin, C. D. Mutto, F. Dominio, L. Minto, and G. M. Cortelazzo, *Time-of-Flight and Structured Light Depth Cameras*. Springer International Publishing, 2016. [Online]. Available: https://doi.org/10.1007/978-3-319-30973-6

[108] G. Agresti, H. Schafer, P. Sartor, Y. Incesu, and P. Zanuttigh, "Unsupervised Domain Adaptation of Deep Networks for ToF Depth Refinement," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[109] M. W. K. Nathan A. Hagen, "Review of snapshot spectral imaging technologies," *Optical Engineering*, vol. 52, no. 9, 2013.

[110] R. Kawakami, Y. Matsushita, J. Wright, M. Ben-Ezra, Y. Tai, and K. Ikeuchi, "High-resolution hyperspectral imaging via matrix factorization," in *CVPR 2011*, 2011, pp. 2329–2336.

[111] J. Aeschbacher, J. Wu, and R. Timofte, "In defense of shallow learned spectral reconstruction from rgb images," in *Proceedings of International Conference on Computer Vision (ICCV) Workshops*, 2017.

[112] S. Galliani, C. Lanaras, D. Marmanis, E. Baltsavias, and K. Schindler, "Learned spectral superresolution," *CoRR*, vol. abs/1703.09470, 2017.

[113] T. Stiebel, S. Koppers, P. Seltsam, and D. Merhof, "Reconstructing spectral images from rgb-images using a convolutional neural network," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[114] Z. Xiong, Z. Shi, H. Li, L. Wang, D. Liu, and F. Wu, "Hscnn: Cnn-based hyperspectral image recovery from spectrally undersampled projections," in *Proceedings of International Conference on Computer Vision (ICCV)Workshops*, 2017.

[115] Y. Yan, L. Zhang, J. Li, W. Wei, and Y. Zhang, "Accurate spectral super-resolution from single rgb image using multi-scale cnn," in *Pattern Recognition and Computer Vision*, J.-H. Lai, C.-L. Liu, X. Chen, J. Zhou, T. Tan, N. Zheng, and H. Zha, Eds. Cham: Springer International Publishing, 2018, pp. 206–217.

[116] B. Arad, O. Ben-Shahar, and R. Timofte, "Ntire 2018 challenge on spectral reconstruction from rgb images," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018.

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

[118] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.

[119] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc., 2014, pp. 2672–2680.

[120] Z. Shi, C. Chen, Z. Xiong, D. Liu, and F. Wu, "Hscnn+: Advanced cnn-based hyperspectral recovery from rgb images," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018.

[121] Y. B. Can and R. Timofte, "An efficient CNN for spectral reconstruction from RGB images," *CoRR*, vol. abs/1804.04647, 2018.

[122] R. M. H. Nguyen, D. K. Prasad, and M. S. Brown, "Training-based spectral reconstruction from a single rgb image," in *Proceedings of European Conference on Computer Vision (ECCV)*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 186–201.

[123] F. Yasuma, T. Mitsunaga, D. Iso, and S. K. Nayar, "Generalized assorted pixel camera: Postcapture control of resolution, dynamic range, and spectrum," *IEEE Transactions on Image Processing*, vol. 19, no. 9, pp. 2241–2253, 2010.

[124] J. Kim, J. Kwon Lee, and K. Mu Lee, "Accurate image super-resolution using very deep convolutional networks," in *IEEE Int. Conference on Computer Vision and Pattern Recognition*, 2016.

[125] H. Chen, J. Gu, O. Gallo, M. Liu, A. Veeraraghavan, and J. Kautz, "Reblur2deblur: Deblurring videos via self-supervised learning," in *IEEE International Conference on Computational Photography (ICCP)*, 2018, pp. 1–9.

[126] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior," in *Conference on Computer Vision and Pattern Recognition*, 2018.

[127] Y. Gandelsman, A. Shocher, and M. Irani, ""double-dip": Unsupervised image decomposition via coupled deep-image-priors," in *IEEE Int. Conference on Computer Vision and Pattern Recognition*, 2019.

[128] Z. Li and N. Snavely, "Learning intrinsic image decomposition from watching the world," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[129] A. Chambolle, "An algorithm for total variation minimization and applications," *Journal of Mathematical Imaging and Vision*, vol. 20, no. 1, pp. 89–97, 2004.

[130] J. P. S. Parkkinen, J. Hallikainen, and T. Jaaskelainen, "Characteristic spectra of munsell colors," *Journal of the Optical Society of America A*, vol. 6, no. 2, pp. 318–322, 1989.

[131] J. Y. Hardeberg, "On the spectral dimensionality of object colours," *Conference on Colour in Graphics, Imaging, and Vision*, vol. 2002, no. 1, pp. 480–485, 2002.

[132] A. Chakrabarti and T. Zickler, "Statistics of Real-World Hyperspectral Images," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 193–200.

[133] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proc. of the Annu. ACM-SIAM Symp. on Discrete Algorithms*, vol. 8, 2007, pp. 1027–1035.

[134] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.

Acknowledgments

I would like to first thank my advisor, professor Pietro Zanuttigh, for guiding me through my Ph.D. studies. I also thank Sony Europe for funding my Ph.D. and in particular Gianluca, Henrik and Valerio for all the help and the suggestions they gave me through these three years. Thanks also to Oliver, Yalcin, Martina, Piergiorgio and Francesco for making me feel welcome at the company.

From the university side, I would like to thank Marco, Umberto, Elena, Daniele, Donald, Giulia and Francesco as the years of Ph.D. pass by quicker and in a better way with good friends and a sufficient amount of spritz.

Finally, I would also like to thank my parents, Cinzia and Franco my brothers, Alessio and Andrea, my grandmas Zita and Italia (also known as Lia) and my aunt Sabrina, for all the patience and support during my Ph.D..