applied sciences

MDPI

Article

Depth Completion with Anisotropic Metric, Convolutional Stages, and Infinity Laplacian

Vanel Lazcano [1,*,†] and Felipe Calderero [2,†]

[1] Facultad de Ciencias, Ingeniería y Tecnología, Universidad Mayor, Chile
[2] CPTO @ Ladorian—Nuclio Digital School, 28002 Madrid, Spain; felipecalderero@gmail.com
* Correspondence: vanel.lazcano@umayor.cl
† These authors contributed equally to this work.

Abstract: Depth map estimation is crucial for a wide range of applications. Unfortunately, it often presents missing or unreliable data. The objective of depth completion is to fill in the "holes" in a depth map by propagating the depth information using guidance from other sources of information, such as color. Nowadays, classical image processing methods have been outperformed by deep learning techniques. Nevertheless, these approaches require a significantly large number of images and enormous computing power for training. This fact limits their usability and makes them not the best solution in some resource-constrained environments. Therefore, this paper investigates three simple hybrid models for depth completion. We explore a hybrid pipeline that combines a very efficient and powerful interpolator (infinity Laplacian or AMLE) and a series of convolutional stages. The contributions of this article are (i) the use a Texture+Structuredecomposition as a pre-filter stage; (ii) an objective evaluation with three different approaches using KITTI and NYU_V2 data sets; (iii) the use of an anisotropic metric as a mechanism to improve interpolation; and (iv) the inclusion of an ablation test. The main conclusions of this work are that using an anisotropic metric improves model performance, and the ablation test demonstrates that the model's final stage is a critical component in the pipeline; its suppression leads to an approximate 4% increase in *MSE*. We also show that our model outperforms state-of-the-art alternatives with similar levels of complexity.

Keywords: depth completion; depth map interpolation; infinity Laplacian; convolutional neural networks; Texture+Structure decomposition; anisotropic metric; KITTI data set; NYU_V2 data set

Citation: Lazcano, V.; Calderero, F. Depth Completion with Anisotropic Metric, Convolutional Stages, and Infinity Laplacian. *Appl. Sci.* **2024**, *14*, 4514. https://doi.org/10.3390/app14114514

Academic Editors: Yi Han, Xiaojun Tan and David Fernández-Llorca

Received: 12 October 2023
Revised: 9 April 2024
Accepted: 24 April 2024
Published: 24 May 2024

1. Introduction

Depth data have become a key source of information for a wide variety of applications, such as robotics and augmented reality. Notably, a depth map is an image whose pixels represent the distance from the object's surface to a fixed point in the scene (typically a sensor capturing the data). There are many sensors to acquire depth data, including structured light and RealSense sensors, ToF (Time-of-Flight) cameras, LiDAR (Light Detection and Ranging), and stereo vision. Regardless of the sensor type, depth data are characterized by low quality in terms of missing data, noise, and errors. Missing data are particularly dramatic, since in some cases the depth maps present large areas with no information or "holes". Completing depth information in those cases becomes an extremely challenging interpolation problem.

Hence, we can define the depth completion problem as the problem of interpolating or filling in the missing gaps in a depth image. It was originally tackled using classical interpolation techniques in image processing and later using more sophisticated inpainting methods. These last methods performed well in simple scenarios at high computational costs. Unfortunately, they presented limitations in more complex real-world situations. For that reason, researchers explored other approaches, such as bilateral filtering and non-local means filtering, which leverage local image features and image statistics to propagate

depth information across empty regions. These models achieved significant improvement but still performed poorly in some real-world situations, for instance, in the presence of large holes corresponding to portions of the reference (color) image without significant texture. It was not until the arrival of deep learning that depth completion experienced a breakthrough, leading to high-quality maps. Nevertheless, despite the success of deep learning algorithms, classical approaches remain relevant in some applications, mainly due to their explainability, computational efficiency, and ability to operate on resource-constrained platforms.

In this work, we explore a non-deep-learning approach that provides good-quality depth maps in constrained environments. We propose a simple and flexible model that requires few images for training and can, consequently, run on a laptop with only one GPU; thus, it represents a low-cost implementation alternative. This paper extends the work presented at the 14th International Workshop on Parallel and Distributed Algorithms and Applications, PDAA [1]. The contributions of this manuscript are as follows:

1. We employ a balanced metric, which considers a balance term between spatial and photo-chromatic distance.
2. We evaluate the impact of an additional Texture+Structure decomposition stage.
3. We propose two variations of the previous interpolation model.
4. We apply an ablation test to our model in the NYU_V2 data set.

1.1. Examples of Incomplete Depth Maps

Here, we illustrate some examples of incomplete depth maps (Figure 1) caused either by sensor misinterpretations, estimation errors, or other factors. Some of these errors may be due to occlusions, transparency, or reflections, among other reasons.

(**a**) (**b**) (**c**) (**d**)

Figure 1. Examples of depth maps containing large areas of missing data. (**a**) Reference color image. (**b**) Depth map, holes (in black) due to occlusions. (**c**) Color reference image; a white square indicates reflections. (**d**) Depth map with holes in areas of high reflection due to uncertainty in the stereo-matching process.

1.2. Organization of This Manuscript

In the next section, we present a brief state of the art and discuss relevant approaches in the field. Next, Section 3 presents our approach, including the underlying mathematical model and the data set employed for training and evaluation. Section 4 details the process for estimating the model parameters. Section 5 presents the performance evaluation. Finally, the conclusion is stated in Section 6.

2. State of the Art

Many image interpolation-related problems have used a scene's color information to guide the propagation of some other type of information, such as depth inpainting and image enhancement [2]. Depth completion is not an exception. Similarly, the geometry of color images has been used to drive a diffusion process completing sparse depth maps [3]. A bilateral filter is one of the most classic examples of a guided method. This image-domain filter considers regional color characteristics, such as edges and smooth areas, to slow or accelerate depth propagation, respectively.

More recently, deep learning techniques have taken the lead. For instance, Lu et al. [4] proposed completing a sparse depth image jointly by reconstructing a gray-level scene image, using convolutional neural networks. They evaluated this method using the KITTI [5] data set, which is a publicly available data set and one of the most popular databases for performance evaluation in the field. Another deep learning approach by Yang et al. [6,7] utilizes a CNN-based model to interpolate depth data from a Kinect sensor database.

Another example is the work of Imran et al. [8]. Their proposal focused on preventing the generation of mixed-depth pixels, that is, pixels that are neither background nor foreground, which are typical in object discontinuities. They proposed a new and interesting representation of pixels called Depth Coefficients (DCs). Hence, the loss function used to train the network is based on the DC representation, leading to a depth interpolation that tries to make more accurate decisions on the image edges. Zhang et al. [9] additionally proposed a model to complete depth maps based on an RGB image and a sparse depth map. Using the RGB image and a deep neural network, the authors estimated the normals of the surfaces in the scene. Using this information and the raw depth data acquired by a sensor, the authors minimized a depth estimation error for the whole image. They evaluated their proposal using their own data set. The results show that it outperforms classical models such as TGV [10] and bilateral filters.

In the work of [11], the authors proposed a model to complete depth data. Their proposal used a unified CNN framework incorporating two features: (a) a model considering the relation between normals and surfaces in the depth value, and (b) a model predicting the confidence of the depth data. In this way, this model predicts surface normals, depth, and depth confidence values simultaneously. These estimations are inputs for a refinement module in order to obtain the final depth estimation. Evaluations of the model in the KITTI and NYU_V2 data sets show that the model presents state-of-the-art performance.

The work in [12] presents a non-local depth completion model, which is different from other non-local models because the proposal dynamically selects more informative neighbors. The depth completion is iteratively refined using confidence values assigned to the depth data. The model also incorporates learned affinity to improve the accuracy of depth estimation, avoiding the "mixed depth" in the edges of the objects in the scene. Experiments show that the proposal outperforms DeepLidar and FuseNet [13] in the KITTI data set.

In [14], a CNN model that fuses sparse depth data and color information to complete a depth map is presented. The model uses two approaches: one processes the color image, and the second refines the depth completion. The outputs from both approaches are combined in order to improve the depth completion. Furthermore, the 3D information of the scene is used to improve the depth estimation. The model achieves state-of-the-art performance in the KITTI data set, outperforming contemporary models.

Lin et al. [15] proposed a model using Dynamic Spatial Propagation Networks. This type of network dynamically estimates the parameters of an affinity. The authors proposed a diffusion suppression procedure to stop diffusion close to the edges of the color image. The most relevant feature of this model is its rapid convergence compared to models with similar architectures.

The paper in [16] presents a deep learning model for depth completion that creates content-dependent and spatially variant kernels. The authors proposed a new convolutional factorization to store the large amount of data associated with these kernels' parameters. This new factorization reduces the computational cost of the model. The proposed model outperforms others in the NYU_V2 and KITTI data sets.

Nevertheless, one of the most well-known limitations of deep learning depth completion models is their huge number of parameters (requiring millions of parameters). Researchers are aware of this drawback, and some improvements have been explored in this direction, for instance, in the work of Bai et al. [17]. Their proposal can reduce the number of parameters by up to 96% compared to similar approaches. Thanks to that, the au-

thors can deploy their model on an FPGA, achieving real-time performance (interpolating 11.1 images per second).

Although these are relevant advances, there is still a need to develop more computationally efficient (non-deep-learning) solutions capable of providing relatively high-quality depth maps with much less computational effort. In this paper, we follow this direction, presenting a simple interpolator based on the infinite Laplacian. This model has already been successfully applied in other image interpolation problems, such as the interpolation of optical flow on video in Lazcano et al. [3].

The origins of this interpolator trace back to the foundational work of Caselles et al. [18]. The authors applied an axiomatic approach to data interpolation, stating a set of properties that a well-behaved interpolator should hold. One of the interpolators holding the full set of axioms was the infinity Laplacian.

More recently, Lazcano et al. [19] also proposed a practical implementation of the infinity Laplacian and the biased infinity Laplacian models. One of the most relevant contributions was the proposal of a Euclidean metric that included a spatial and a color term. The infinity Laplacian provided the best results for upsampling images in the context of the Middlebury data set.

In addition to the infinity Laplacian, there is a panoply of computationally efficient techniques in the literature. For instance, in [20], the authors segmented the reference color image using superpixels and the SLIC algorithm. A 2D linear interpolation approach was used to generate a depth-value plane using the depth information in each superpixel. The model incorporated heuristics and simple rules to tackle depth estimation when segmentation errors occurred. The results outperformed the bilateral filtering approach.

A more recent and interesting approach is the work by Saidi et al. [21]. The authors proposed a real-time algorithm capable of estimating up to 111 depth maps per second. Unfortunately, the estimated depth maps presented missing or low-confidence data.

Table 1 summarizes the number of parameters, GPUs, and images in the training set reported in the literature of cited methods.

Table 1. Model training comparison table.

Order Number	Model	Number of Parameters	Number of GPUs	Number of Images in the Training Set
1	Eldesokey et al. [22]	330.000	-	-
2	Lu et al. [4]	-	2	85,898
3	Bai et al. [17]	134.000	1	85,898
4	Lin et al. [15]	-	4	50,000
5	Morpho Networks [23]	-	1	4300
6	Chen et al. [24]	-	8	-

Table 1 shows that most models use thousands of images for training and more than one GPU, and two report thousands of parameters to be estimated.

Last but not least, it is important to mention that some consensus has been reached by the depth completion community regarding performance evaluation. Researchers used well-established data sets, which include clear evaluation protocols, to benchmark their solutions. This article will use two popular data sets: KITTI [5] and NYU_V2 [25].

3. Materials and Methods

In this section, we present the use of the infinity Laplacian, or AMLE (Absolutely Minimizing Lipschitz Extension), for the interpolation of sparse depth maps. We selected the infinity Laplacian because it is (i) simple to solve, (ii) fast to compute, and (iii) easy to implement. As we will explain in Section 3.7, the infinity Laplacian has several parameters, requiring a small training set to estimate them.

We embedded the infinity Laplacian in a pipeline, considering convolutional stages to enforce features or filter noise. In the following subsections, we explain the interpolation model and the pipeline.

3.1. Depth Interpolation Models

We used the infinity Laplacian to interpolate or complete the available depth data. The infinity Laplacian model was initially proposed by Aronsson in the 1960s, and it is the simplest interpolator that satisfies a set of axioms [18]. We will explain the infinity Laplacian in Section 3.7.

3.2. Pipeline

Figure 2 shows the proposed pipeline to complete depth maps. The input to the pipeline is a CIE-Lab reference image. The pipeline includes a feature-extraction convolutional stage, an interpolation stage, and a noise-removal convolutional stage.

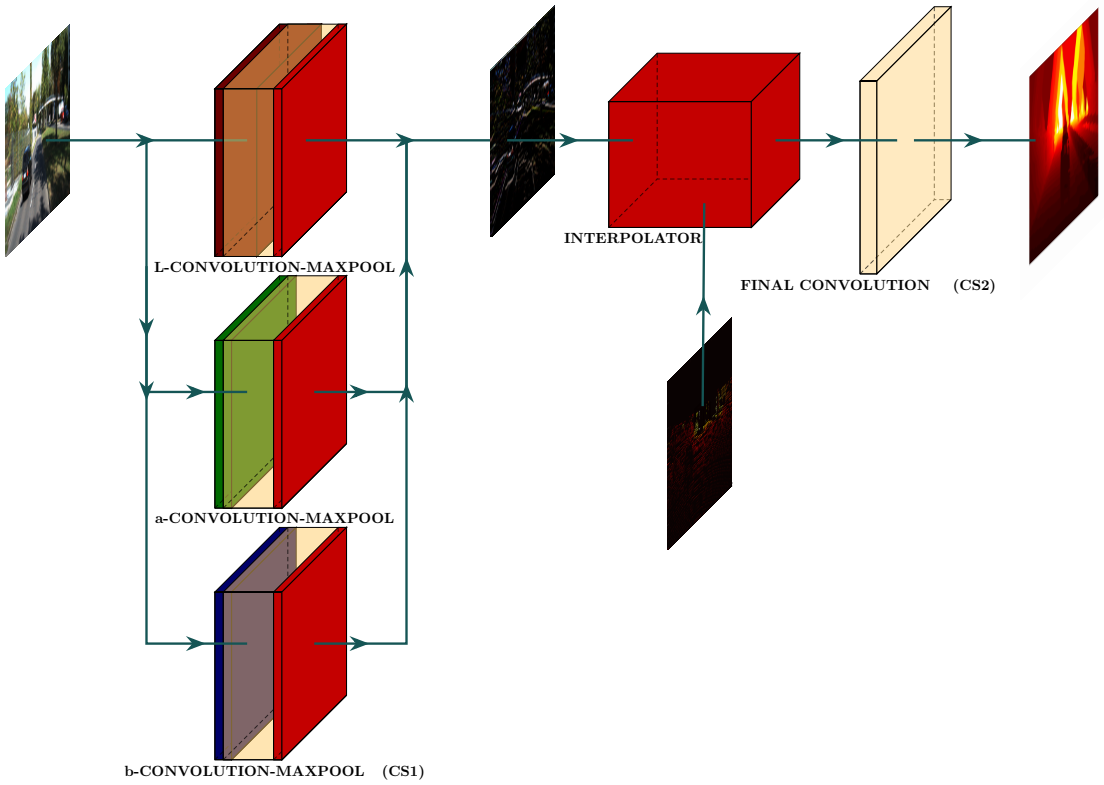


Figure 2. Proposed pipeline structure. It consists of three steps. CS1 extracts color features from the reference image using Gabor filters. Then, the algorithm propagates the acquired available depth data, with the guidance of the color image, to all regions of the sparse depth map by solving the infinity Laplacian equation (interpolator). Finally, we use a convolution stage (CS2) to eliminate outliers and noise from the completed depth map.

3.2.1. Convolutional Stage (CS1)

We processed each color component (L, a, b) of the reference image with a certain number of Gabor filters, $NFilt$ ($NFilt = 1, ..., 8$), to extract the color features. The Gabor filters and their parameters are as follows:

$$\begin{aligned} f_{odd}(x', y') &= e^{\left(-\frac{\|\mathbf{x}\|}{2\sigma^2}\right)} \sin(wx') \\ f_{even}(x', y') &= e^{\left(-\frac{\|\mathbf{x}\|}{2\sigma^2}\right)} \cos(wx'), \end{aligned} \tag{1}$$

where

$$x' = x\cos(\theta) + y\sin(\theta)$$
$$y' = -x\sin(\theta) + y\cos(\theta). \quad (2)$$

$\mathbf{x} = (x', y')$, θ is the rotation angle, and $w = \frac{2\pi}{\lambda}$ is the spatial frequency.

Then, we max-pooled the output of the filters in each color component and normalized the output to the range 0–255. Finally, we concatenated the normalized color components to create a color feature image. Figure 3 shows some examples of processed images.

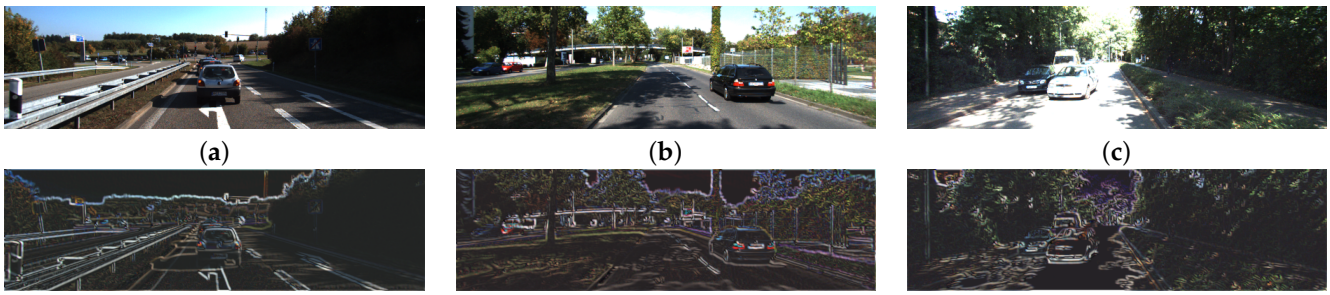


Figure 3. Examples of color reference images processed by the first convolutional stage. Figures (**a**–**c**) show the color reference image extracted from the KITTI data set. In (**d**–**f**), we present their respective output after being processed by the first convolutional stage (color feature images). We observe in (**d**–**f**) that the processing mainly emphasizes horizontal edges in order to improve the diffusion process.

For the examples shown, the first stage enforces horizontal edges due to the characteristics of the images. These images were acquired by a horizontal laser scan using a LiDAR sensor, which is why the horizontal edges were particularly relevant to drive diffusion.

The parameters used for each one of the Gabor filters were estimated through a training process. We will detail this process in Section 4.

3.2.2. Interpolator (Infinity Laplacian)

As shown in Figure 2, the central element of the proposal is the infinity Laplacian interpolator. The infinity Laplacian takes two inputs: a color reference image and a sparse depth map. The model interpolates the available depth data, guided by the color reference image. This interpolation is achieved by solving the degenerated second-order partial differential equation, the infinity Laplacian, using the available depth data. The solution of the infinity Laplacian propagates the available data to the empty regions. This propagation is performed through a diffusion process based on an anisotropic metric, which is explained in Section 3.7.

3.3. Final Convolutional Stage (CS2)

In this stage, we process the output of the infinity Laplacian by a bank of average filters, eliminating noise and outliers. The eight parameters of the filters are estimated in the training stage (see Section 4).

3.4. Pipeline with a Variable Structure

The proposed pipeline can dynamically change the order in which the stages process the images (processing sequence). As mentioned earlier, the order depicted in Figure 2 is CS1–Interpolator–CS2, but depending on a boolean parameter, the processing sequence can be CS2–Interpolator–CS1. The dimensions and number of filters in each stage are adjusted to allow the processing sequence to be inverted. The idea behind this operation is that the structure of the model switches during training. Once the parameters of the model are

determined, they are set into the pipeline, and after that, the model is used to process the validation set.

3.5. Variable Number of Filters

The Gabor and average filters in our proposal are variable, ranging from 1 to 8 and 1 to 9, respectively. Only two parameters define the number of used filters ($NFilt$ and $NFilt2$). These parameters are estimated during the training and set after that.

3.6. Texture+Structure Decomposition

An image $I(\mathbf{x})$ can be decomposed into its structural ($I_S(\mathbf{x})$) and textural ($I_T(\mathbf{x})$) components using the Rudin–Osher–Fatemi (ROF) total variation denoising model. The structural component is obtained by solving for the intensity values of the image $I(\mathbf{x})$:

$$\min_{I_S} \int_\Omega \left\{ (|\nabla I_S(\mathbf{x})| + \frac{1}{2\theta}|I_S(\mathbf{x}) - I(\mathbf{x})|^2 \right\} d\mathbf{x} \tag{3}$$

where $I(\mathbf{x})$ is the original image, $\Omega \subset \mathbb{R}^2$ its domain, and $\theta \in \mathbb{R}$. We remark that $I_T(\mathbf{x}) = I(\mathbf{x}) - I_S(\mathbf{x}) \quad \forall \mathbf{x} \in \Omega$.

We used the Texture+Structure decomposition to pre-process the color reference image. This way, the texture components (high spatial variations) and the structure of the image (low spatial variations) were extracted from the reference image. Figure 4 shows an example of the Texture+Structure decomposition.

(**a**) (**b**) (**c**)

Figure 4. Example of Texture+Structure decomposition. (**a**) Original color reference image. (**b**) Low spatial variation part of the image. (**c**) High spatial variation part of the image (texture component).

In Figure 4a, we show the original image; in Figure 4b, the structure component; and in Figure 4c, the texture component.

3.7. Infinity Laplacian

The interpolator used to complete the sparse depth maps is the infinity Laplacian, or AMLEhows. The infinity Laplacian is the simplest model that holds a set of axioms for interpolators [18]. Suppose $\Omega \subset \mathbb{R}^2$ is the image domain. For each incomplete depth map u, we solve the following problem:

$$\Delta_{\infty,g} u = 0 \in \Omega, \tag{4}$$

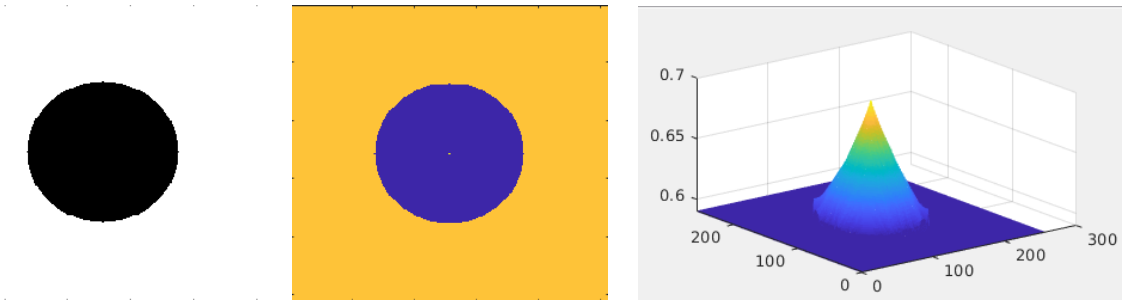
where $\Delta_{\infty,g} u$ is the infinity Laplacian, g is the metric, and u is the interpolated depth data. Let us consider the image domain Ω as a rectangle, let Ω be a domain in a smooth compact two-dimensional manifold $\mathcal{M}$ embedded in $\mathbb{R}^3$. The interpolated surface u, or manifold, is obtained by solving the infinity Laplacian, Equation (4), in the Ω domain. Considering solving the infinity Laplacian in this 2D surface $\mathcal{M}$, the infinity Laplacian is stated as

$$\Delta_{\infty,g} u := D^2_{\mathcal{M}} u \left(\frac{\nabla u}{|\nabla u|}, \frac{\nabla u}{|\nabla u|} \right), \tag{5}$$

which is a degenerated elliptic PDE. Let us consider the depth map u as $u : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$, and the available data are located in $\mathcal{O} \subset \Omega$, i.e., $u|_{\partial\mathcal{O}} = \theta$. Considering the boundary condition for the infinity Laplacian, we set the available depth data to a constant value θ on

the domain's boundary. This fact ensures that the completed depth map agrees with the available data at the boundary.

As a proof of concept, we will show an example of data interpolation using the infinity Laplacian. Figure 5 shows a simple example of depth completion using the infinity Laplacian.

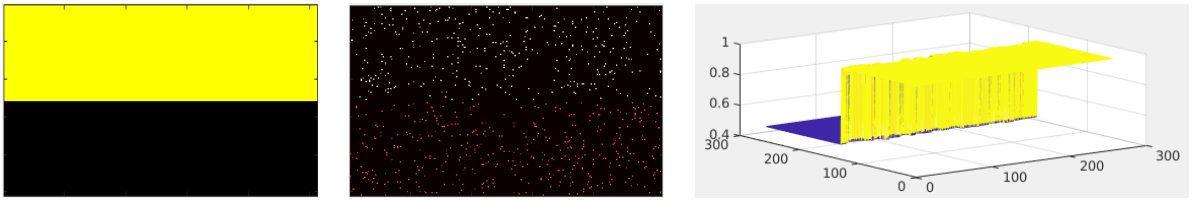


(**a**) (**b**) (**c**)

Figure 5. Example of depth completion by the infinite Laplacian. We show the reference image in (**a**). The image consists of a white rectangle, and in its center, there is a black circle. We show the depth data in (**b**). The orange region represents a constant depth of 0.5; in the center, the circle in blue represents the lack of data. In the center of the blue circle, there is only a single value equal to 0.68. We show the interpolated depth map in (**c**). The infinity Laplacian generates a cone, connecting the circular contour with the point in the circle's center.

Figure 5 shows a depth map in (**a**) with a circular hole to be completed. Furthermore, the circle has a unique depth data point in its center. Let $\vec{0}$ be the coordinates of the depth data point in the center of the circle, and let $\mathcal{C}$ be the circular contour of radius 60 centered at $\vec{0}$. Then, we state that $u\big|_{\vec{0}} = 0.68$ and $u\big|_{\mathcal{C}} = 0.5$. We solve the infinity Laplacian in Equation (4), obtaining the cone shown in Figure 5c.

In Figure 6, we show a second simple example where we complete a two-level depth image given random depth samples.



(**a**) (**b**) (**c**)

Figure 6. Example of depth map completion. (**a**) shows a two-color reference image. (**b**) shows random samples of a two-level depth map. White values represent the value 1, and orange values represent the depth value 0.5. Figure (**c**) shows a 3D representation of the completed depth map. We observe the two-level surface.

Figure 6a shows the reference color image of this second toy example. Figure 6b shows the sampled depth map. We sampled 2% of the depth data image, which is around 1300 samples. The completed two-level depth data are plotted in 3D in Figure 6c.

Equation (4) was discovered by Aronsson in the nineteen-sixties as an Euler–Lagrange equation for the problem of the Absolutely Minimizing Lipschitz Extension (or AMLE):

$$\min_{u} \lim_{p\to\infty} \left\{ \int_{\Omega} |\nabla u|^p d\mathbf{x} \right\}, \tag{6}$$

where u is the extension of the available data. Computing the Euler–Lagrange equation of Equation (6), we have

$$\nabla \cdot (\|\nabla u(\mathbf{x})\|^{p-2} \nabla u(\mathbf{x})) = 0, \tag{7}$$

that is to say,

$$\|\nabla u(\mathbf{x})\|^{p-2}(\Delta u(\mathbf{x}) + (p-2)\Delta_\infty u(\mathbf{x})) = 0, \tag{8}$$

and when $p \to \infty$, we have

$$\Delta_\infty u(\mathbf{x}) = 0, \tag{9}$$

where $\Delta_\infty u(\mathbf{x})$ is the Laplacian of u computed in the gradient direction.

The infinity Laplacian model enables the creation of cones and soft surfaces, which could be more suitable for completing urban scenes.

3.8. Variations in the Infinity Laplacian

The biased infinity Laplacian is a variation of the original in Equation (9) used in [19]. This operator additionally considers the modulus of the gradient of u:

$$\Delta_\infty u(\mathbf{x}) + c|\nabla u(\mathbf{x})| = 0, \tag{10}$$

where the c parameter is a positive real value. For the value $c = 0$, we recover the infinity Laplacian. We state variations in the biased infinity Laplacian, namely the balanced biased infinity Laplacian adding a balance term $\gamma(\mathbf{x})$:

$$\gamma(\mathbf{x})\Delta_\infty u(\mathbf{x}) + (1-\gamma(\mathbf{x}))c|\nabla u(\mathbf{x})| = 0, \tag{11}$$

The balance term $\gamma : \Omega \subset \mathbb{R}^2 \to [0,1]$ can be explicitly computed as

$$\gamma(x) = \frac{1}{1 + e^{\beta_\gamma(|\Delta_\infty u(\mathbf{x})| - \tau_\gamma|\nabla u(\mathbf{x})|)}}, \tag{12}$$

where β_γ and τ_γ are real positive parameters that will be estimated in the training stage. $\gamma(\mathbf{x})$ is a balanced map $\forall \mathbf{x} \in \Omega$; depending on the necessity of the completion process, the model balances between infinity Laplacian and the eikonal operator. A second variation of this proposal is called the double balanced infinity Laplacian, which considers the infinity Laplacian using the eikonal operators in Equation (11):

$$\frac{\gamma(\mathbf{x})}{2}\left(\phi(\mathbf{x})\|\nabla u(\mathbf{x})\|_\xi^+ + (1-\phi(\mathbf{x}))\|\nabla u(\mathbf{x})\|_\xi^-\right) + (1-\gamma(\mathbf{x}))c|\nabla u(\mathbf{x})| = 0, \tag{13}$$

where $\phi(.)$ is a similar mechanism to $\gamma(\mathbf{x})$, defined as

$$\phi(\mathbf{x}) = \frac{1}{1 + e^{\beta_\phi(|\|\nabla u(\mathbf{x})\|_\xi^+| - \tau_\phi|\|\nabla u(\mathbf{x})\|_\xi^-|)}}, \tag{14}$$

with β_ϕ and $\tau_\phi \in \mathbb{R}^+$. We remark that if $\phi(\mathbf{x}) = \gamma(\mathbf{x}) = 0.5$, we recover the biased infinity Laplacian.

3.9. Metric

Let us consider the reference color image as a function from the Ω domain to the color space, i.e., $I : \Omega \subset \mathbb{R}^2 \to \mathbb{R}^3$, and the metric $d_{\mathbf{xy}} : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$, conforming to the manifold $\mathcal{M} = (\Omega, d_{\mathbf{xy}})$. Considering $\mathcal{M}$, we solve Equation (4). We explored different metrics to estimate the shape of the manifold in order to better approximate the geodesic distance. In a preliminary work [19], we tested the following metric:

$$d_{\mathbf{x},\mathbf{y}} = \kappa_x\|\mathbf{x}-\mathbf{y}\|^2 + \kappa_c \sum_{i \in L,a,b} |I(\mathbf{x}) - I(\mathbf{y})|^2.$$

where the constants κ_x and $\kappa_c \in \mathbb{R}$ are estimated in the training stage. This metric comprises two terms: a spatial distance term $\|\mathbf{x}-\mathbf{y}\|^2$ and a photo-chromatic distance term $\|I(\mathbf{x}) - I(\mathbf{y})\|^2$. The first term is the spatial distance of pixels $\mathbf{x}$ and $\mathbf{y}$ in the image domain Ω.

The second term is the photo-chromatic distance in the CIE-Lab color space between $I(\mathbf{x})$ and $I(\mathbf{y})$. To give more flexibility to this metric, we relaxed the exponents as follows:

$$d_{\mathbf{x},\mathbf{y}} = \left[\kappa_x \|\mathbf{x}-\mathbf{y}\|^s + \kappa_c \sum_{i\in L,a,b} |I(\mathbf{x}) - I(\mathbf{y})|^p\right]^q$$

where $p, q, s \in \mathbb{R}^+$. We also estimated the exponent values in the training stage.

We stated (in our work presented in [1]) the following metric:

$$d_{\mathbf{x},\mathbf{y}} = \left(\kappa_x[(\mathbf{x}-\mathbf{y})^T A(\mathbf{x}-\mathbf{y})]^s + \kappa_c\left[(I(\mathbf{x})-I(\mathbf{y}))^T C(I(\mathbf{x})-I(\mathbf{y}))\right]^p\right)^q, \quad (15)$$

where A and C are positive definite matrices, and $d_{\mathbf{x},\mathbf{y}}$ is called the positive definite metric.

In this work, we propose an anisotropic metric, which considers a balance term $\theta(\mathbf{x})$ between spatial distance and photo-chromatic distance,

$$d_{\mathbf{x},\mathbf{y}} = \left[\kappa_x\theta(\mathbf{x})[(\mathbf{x}-\mathbf{y})^T A(\mathbf{x}-\mathbf{y})]^s + \kappa_c(1-\theta(\mathbf{x}))\left[(I(\mathbf{x})-I(\mathbf{y}))^T C(I(\mathbf{x})-I(\mathbf{y}))\right]^p\right]^q, \quad (16)$$

where $\theta(\mathbf{x})$ is defined by

$$\theta(\mathbf{x}) = \frac{1}{1+e^{\beta_\theta(D_\sigma(\mathbf{x})-\tau_\theta D_\phi(\mathbf{x}))}},$$

where $D_\sigma(\mathbf{x}) = \|\mathbf{x}-\mathbf{y}\|^s$ is the spatial distance, $D_\phi(\mathbf{x}) = \log\left(1+\sum_{i\in L,a,b}|I(\mathbf{x})-I(\mathbf{y})|^p\right)$ is the photo-chromatic distance, and $\beta_\theta, \tau_\theta \in \mathbb{R}^+$. On the one hand, if $D_\sigma(\mathbf{x}) \gg D_\phi(\mathbf{x})$, the difference between $D_\sigma(\mathbf{x}) - \tau_\theta D_\phi(\mathbf{x})$ should be large, and positive. It means that $e^{\beta_\theta(D_\sigma(\mathbf{x})-\tau_\theta D_\phi(\mathbf{x}))}$ should be also large, and finally $\theta(\mathbf{x})$ should be small, i.e., $\theta(\mathbf{x}) \approx 0$. The metric is more confident in the photo-chromatic term.

On the other hand, if $D_\sigma(\mathbf{x}) \ll D_\phi(\mathbf{x})$, the difference between $D_\sigma(\mathbf{x}) - \tau_\theta D_\phi(\mathbf{x})$ should be negative. It means that $e^{\beta_\theta(D_\sigma(\mathbf{x})-\tau_\theta D_\phi(\mathbf{x}))}$ should be small, ≈ 0, that is to say, that $\frac{1}{1+e^{\beta_\theta(D_\sigma(\mathbf{x})-\tau_\theta D_\phi(\mathbf{x}))}} \approx 1$, meaning that the metric is more confident in the spatial term. As a proof of concept of $\theta(\mathbf{x})$, we show estimation examples of the balance term considering a color reference image in Figure 7.

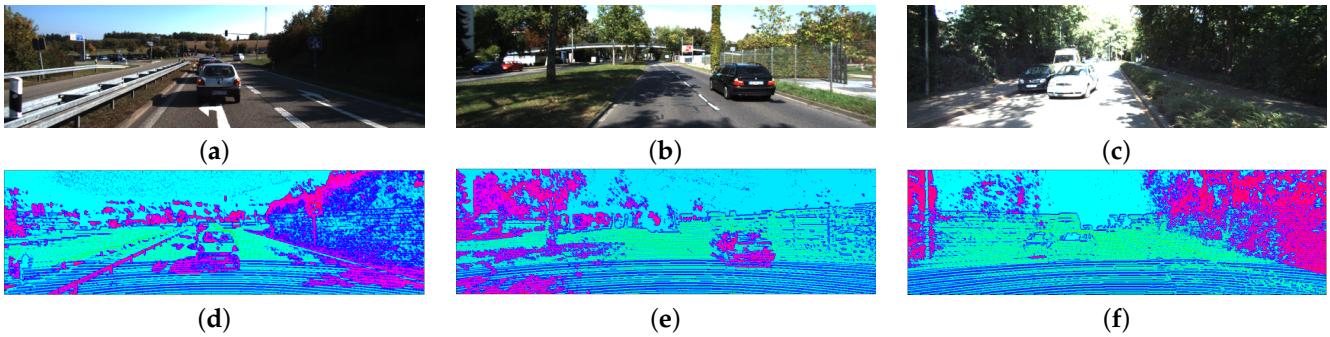


Figure 7. Example of $\theta(\mathbf{x})$ map in three example images. Figures (**a**–**c**) show considered reference color images. Figures (**d**–**f**) show color-coded values of $\theta(\mathbf{x})$ balance map. In magenta, we show largest values of $\theta(\mathbf{x})$ ($0.60 \leq \theta(\mathbf{x}) \leq 0.7$). In blue and cyan, we show intermediate values of $0.43 < \theta(\mathbf{x}) < 0.60$. In green, we show lowest values of $\theta(\mathbf{x})$ ($0.33 < \theta(\mathbf{x}) < 0.43$).

Figure 7 shows color-coded values of the $\theta(\mathbf{x})$ map. Figure 7a–c are reference color images already shown in Figure 3. In Figure 7d–f, we color-coded the values of the map $\theta(\mathbf{x})$. The metric is more confident in the spatial term in magenta regions. Magenta regions represent objects with shadows and highly textured regions. In blue, we represent $0.43 < \theta(\mathbf{x}) < 0.6$, which means that the metric is confident in the spatial term. We can see

that the $\theta(\mathbf{x})$ values are less than 0.5 in cyan areas, which is the sky in the reference color image, which means that the metric is confident of both the photo-chromatic and spatial term. In light-green regions, $\theta(\mathbf{x})$ has the smallest values, which means that the metric is more confident in the photo-chromatic term.

3.10. Geodesic Metric Approximation

Let us consider the discrete image domain $\Omega \subset \mathbb{R}^2$ as a grid, and let $\mathbf{x}$ and $\mathbf{y}$ be two points in the grid. We defined a metric $d_{\mathbf{xy}}$ in Equation (16) between consecutive points. The geodesic distance between two points in the grid represents the shortest path joining $\mathbf{x}$ and $\mathbf{y}$, i.e.,

$$d_g(\mathbf{x}, \mathbf{y}) = \min_L \{L : \text{the length of the path joining } \mathbf{x} \text{ and } \mathbf{y}\}, \tag{17}$$

that is to say, considering a curve $\{\gamma(i)\}_{i=0}^m$ joining $\mathbf{x}$ and $\mathbf{y}$ in the grid, the length of γ is given by

$$L_g(\gamma) = \sum_{i=0}^{m} d_{\gamma(i),\gamma(i+1)}. \tag{18}$$

The geodesic distance $d_g(\mathbf{x}, \mathbf{y})$ should be computed using Dijkstra's algorithm. To increase the efficiency of computing the geodesic distance, we approximated the geodesic distance directly with the following metric:

$$d_g(\mathbf{x}, \mathbf{y}) \approx d_{\mathbf{xy}}.$$

This approach offers efficient computation while keeping an acceptable level of approximation.

3.11. Solving the AMLE Model

For the sake of completeness, we briefly explain the numerical solution of the infinity Laplacian model as already presented in [26,27]. We remind the reader that the infinity Laplacian is equivalent to

$$\Delta_{\infty,g} u = \frac{1}{2}\left(\|\nabla u(\mathbf{x})\|_x^+ + \|\nabla u(\mathbf{x})\|_x^-\right) = 0. \tag{19}$$

where $\|\nabla u(\mathbf{x})\|_x^+$ and $\|\nabla u(\mathbf{x})\|_x^-$ are the positive eikonal operator and negative eikonal operator, respectively.

3.12. Practical Model Implementation

Given a point $\mathbf{x}$ in a neighborhood $\mathcal{N}(\mathbf{x})$ of $\mathbf{x}$, the positive eikonal operator is defined as

$$\|\nabla u(\mathbf{x})\|_x^+ = \max_{\mathbf{y} \in \mathcal{N}(\mathbf{x})} \frac{u(\mathbf{y}) - u(\mathbf{x})}{d_{\mathbf{xz}}}, \tag{20}$$

and the negative eikonal operator is defined as

$$\|\nabla u(\mathbf{x})\|_x^- = \min_{\mathbf{z} \in \mathcal{N}(\mathbf{x})} \frac{u(\mathbf{z}) - u(\mathbf{x})}{d_{\mathbf{xz}}}. \tag{21}$$

Let $\mathbf{y}$ and $\mathbf{z}$ be the locations that maximize Equations (20) and (21), respectively. With this definition, it is possible to state the infinity Laplacian,

$$\frac{1}{2}\left(\left(\frac{u(\mathbf{y}) - u(\mathbf{x})}{d_{\mathbf{xy}}}\right) + \left(\frac{u(\mathbf{z}) - u(\mathbf{x})}{d_{\mathbf{xz}}}\right)\right) = 0. \tag{22}$$

Solving for $u(\mathbf{x})$,

$$u(\mathbf{x}) = \frac{d_{\mathbf{xz}} u(\mathbf{y}) + d_{\mathbf{xy}} u(\mathbf{z})}{d_{\mathbf{xz}} + d_{\mathbf{xy}}}. \tag{23}$$

The iterated version is

$$u^{k+1}(\mathbf{x}) = \frac{d_{\mathbf{xz}}u^k(\mathbf{y}) + d_{\mathbf{xy}}u^k(\mathbf{z})}{d_{\mathbf{xz}} + d_{\mathbf{xy}}} \tag{24}$$

with $k \in \mathbb{N} \cup \{0\}$.

3.13. Numerical Model for the Biased Infinity Laplacian

Taking into account the above definitions (of the biased infinity Laplacian) and substituting in Equation (10), we have

$$\frac{1}{2}\left(\frac{u(\mathbf{y}) - u(\mathbf{x})}{d_{\mathbf{xy}}} + \frac{u(\mathbf{z}) - u(\mathbf{x})}{d_{\mathbf{xz}}}\right) + c\left|\frac{u(\mathbf{y}) - u(\mathbf{x})}{d_{\mathbf{xy}}}\right| = 0, \tag{25}$$

which is a linear equation for $u(\mathbf{x})$; consequently, we can solve for $u(\mathbf{x})$:

$$u(\mathbf{x}) = \frac{(1 + 2c\,\mathrm{sgn}(u(\mathbf{y}) - u(\mathbf{x})))u(\mathbf{y})d_{\mathbf{xz}} + u(\mathbf{z})d_{\mathbf{xy}}}{(1 + 2c\,\mathrm{sgn}(u(\mathbf{y}) - u(\mathbf{x})))u(\mathbf{y}) + u(\mathbf{z})}. \tag{26}$$

As shown in Equation (26), the solution of the biased infinity Laplacian is only the weighted sum of the $u(\mathbf{y})$ and $u(\mathbf{z})$ values multiplied by the distance of the center of $\mathcal{N}(\mathbf{x})$ to the points $\mathbf{y}$ and $\mathbf{z}$, respectively. We remark that

$$|u(\mathbf{x}) - u(\mathbf{y})| = \mathrm{sgn}(u(\mathbf{x}) - u(\mathbf{y}))(u(\mathbf{x}) - u(\mathbf{y})).$$

where $\mathrm{sgn}()$ is the sign function.

3.14. Numerical Model for the Balanced Biased Infinity Laplacian

Recalling the balanced infinity Laplacian, we substitute it into Equation (11) to obtain

$$\frac{\gamma(\mathbf{x})}{2}\left(\frac{u(\mathbf{y}) - u(\mathbf{x})}{d_{\mathbf{xy}}} + \frac{u(\mathbf{z}) - u(\mathbf{x})}{d_{\mathbf{xz}}}\right) + (1 - \gamma(\mathbf{x}))c\left|\frac{u(\mathbf{y}) - u(\mathbf{x})}{d_{\mathbf{xy}}}\right| = 0. \tag{27}$$

We can solve for $u(\mathbf{x})$:

$$u(\mathbf{x}) = \frac{(1 + 2c(1 - \gamma(\mathbf{x}))\,\mathrm{sgn}(u(\mathbf{y}) - u(\mathbf{x})))u(\mathbf{y})d_{\mathbf{xz}} + \gamma(\mathbf{x})u(\mathbf{z})d_{\mathbf{xy}}}{(1 + 2c(1 - \gamma(\mathbf{x}))\,\mathrm{sgn}(u(\mathbf{y}) - u(\mathbf{x})))u(\mathbf{y}) + \gamma(\mathbf{x})u(\mathbf{z})}. \tag{28}$$

3.15. Numerical Model for the Double Balanced Biased Infinity Laplacian

Taking into account the above definition and substituting it into Equation (11), we have

$$\frac{\gamma(\mathbf{x})}{2}\left(\phi(\mathbf{x})\left[\frac{u(\mathbf{y}) - u(\mathbf{x})}{d_{\mathbf{xy}}}\right] + (1 - \phi(\mathbf{x}))\left[\frac{u(\mathbf{z}) - u(\mathbf{x})}{d_{\mathbf{xz}}}\right]\right) + (1 - \gamma(\mathbf{x}))c\left|\frac{u(\mathbf{y}) - u(\mathbf{x})}{d_{\mathbf{xy}}}\right| = 0. \tag{29}$$

Solving for $u(\mathbf{x})$,

$$u(\mathbf{x}) = \frac{(\phi(\mathbf{x}) + 2c(1 - \gamma(\mathbf{x}))\,\mathrm{sgn}(u(\mathbf{y}) - u(\mathbf{x})))u(\mathbf{y})d_{\mathbf{xz}} + (1 - \phi(\mathbf{x}))\gamma(\mathbf{x})u(\mathbf{z})d_{\mathbf{xy}}}{(\phi(\mathbf{x}) + 2c(1 - \gamma(\mathbf{x}))\,\mathrm{sgn}(u(\mathbf{y}) - u(\mathbf{x})))u(\mathbf{y}) + (1 - \phi(\mathbf{x}))\gamma(\mathbf{x})u(\mathbf{z})}. \tag{30}$$

3.16. Iterative Solution of the Infinity Laplacian Variations

Given an initial solution $u^0(\mathbf{x})$, we can compute in each neighborhood $\mathcal{N}(\mathbf{x})$ the values of $u(\mathbf{y})$ and $u(\mathbf{z})$. Then, we iteratively update $u(\mathbf{x})$ using any of the expressions in Equations (26), (28), or (30), depending on the selected model.

3.17. Data Sets

In this work, we use two publicly available data sets, (1) KITTI and (2) NYU_V2, to train and validate the proposed models.

1. KITTI depth Completion Suite [5] consists of 1000 color images and depth data of urban scenes acquired by a color camera and a LiDAR (Light Detection and Ranging), respectively, mounted on a vehicle traveling across a city. The vehicle had a color camera on its top, and as the vehicle traveled across the city, its sensors acquired synchronized data. Each color image is accompanied by a corresponding sparse depth map and ground truth, as illustrated in Figure 8.

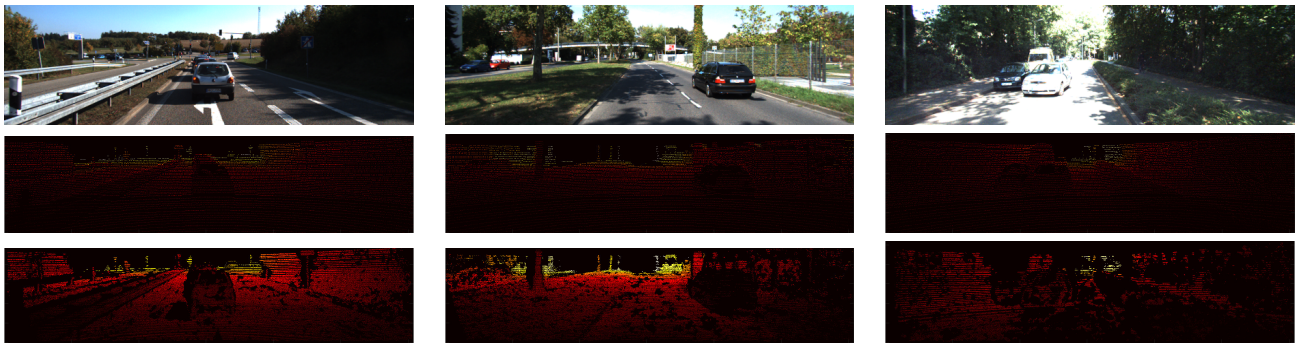


Figure 8. Examples of reference color images of the KITTI data set, sparse depth maps, and ground truth. First row: color reference images. Second row: available sparse depth maps. In the third row, we show the corresponding ground truth. We color-coded the available depth value using a MATLAB jet color map. In the second and third rows, black means a lack of data, and red and yellow mean small and large depth values, respectively.

The model was trained using three images extracted from the KITTI data set, with their corresponding depth maps and their corresponding ground truth. We used the 997 remaining images to evaluate the model's performance.

2. NYU_V2 data set
The NYU_V2 data set comprises 1449 indoor images and their corresponding depth data acquired by a structure light sensor. In Figure 9, we show some images of NYU_V2.

Figure 9. Examples of NYU_V2 data set. The first row shows three indoor color images of the NYU_V2 data set. We show images Img_1199, Img_1372, and Img_1424, respectively. We color-coded the corresponding depth map in the second row using MATLAB jet colormap: dark red means small depth values, and bright yellow means large depth values.

3. Similarly to the previous data set, three images from NYU_V2 were used to train the model. Images from 1001 to 1449 were used to estimate the model's performance (as stated in the corresponding protocol).

3.18. Implementation and Complexity

We implemented the proposed model in C++/Cuda and used MATLAB to implement the PSO algorithm and to plot the results. The program runs in Linux 22.04 on a laptop with HP OMEN 16, 16 GB of RAM, a Processor i7-11800H, and an NVIDIA GTX 3070. We measured the processing time. Our computer processed each image with $NIter = 79$ iterations, obtaining a running time of 0.08 ms per iteration. This processing time means that the total processing time is 0.64 ms. Please note that the time to acquire and store the image was not included. Given the above factors, the processing time can be ≈ 1 ms.

In terms of complexity, every time we interpolated an image, it was processed pixel-by-pixel. Let N be the number of image pixels, and consider a neighborhood of P pixels around each pixel x. The central pixel is compared with every pixel of the neighborhood, leading to

$$\text{number of operation} = O(N \times P)$$

The size of the neighborhood around x is presented in Table 2:

Table 2. Number of points per neighborhood size.

Order Number	Neighborhood Size	Parameter P
1	3×3	9
2	5×5	25
3	7×7	49
4	9×9	81
5	11×11	121
6	13×13	169

For an image of $N = 1216 \times 352 = 428032$ and a logarithm of $log(N) = 12.96$, if we keep $1 \leq radius \leq 2$, we have $O(Nlog(N))$ operations.

In Table 3, we report the processing time of our model and other models considered in this paper.

Table 3. Running time of different models.

Model	Processing Time
Our model	1 ms
Eldesokey et al. [22]	20 ms
Lu et al. [4]	50 ms
Bai et al. [17]	90 ms
Imran et al. [8]	150 ms
Lin et al. [15]	160 ms
Morpho. Networks [23]	170 ms
Krauss et al. [20]	840 ms
Conv. Spatial. Prop. Networks [24]	1000 ms

Table 3 shows that our model is approximately an order of magnitude faster than the other models.

4. Training the Model

To estimate the model parameters, the PSO (Particle Swarm Optimization) algorithm was applied. In Table 4, we summarize the total number and values of the model parameters:

Table 4. Parameters of the proposed model.

Parameter	Description	Number of Parameters
κ_x, κ_c, p, q, s, β_α, τ_α	Parameters of the metric	7
β_θ, τ_θ	Parameters of the anisotropic metric	2
radius	Neighborhood size of $\mathcal{N}(\mathbf{x})$	1
n_{iter}	Number of iterations	1
σ_j, w_j with j = 1,...,8	Parameters of the Gabor Filters CS1	16
$w_{1j}, w_{2j}, w_{3j}, w_{4j}$ with $j = 1, ..., 9$	Weights of the convolutional filters stage CS2	36
a_{11}, a_{12} and c_{11}, c_{12}	Coefficients of the matrices A and C	4
Reversible	Sequence order	1
NFilt, NFilt2	Number of filters considered in stage CS1 and CS2, respectively	2
β_γ, τ_γ	Parameters of the balanced infinity Laplacian	2
β_ϕ, τ_ϕ	Parameters of the double balanced infinity Laplacian	2
Total		74

Each individual in the PSO method is a candidate solution to minimize the Depth Completion Error, given by 50 random instances of $\mu_i \in \mathbb{R}^{74}$ ($i = 1, ..., 50$). The error formula is given by

$$J(\mu_i) = \sum_{j=1}^{N} MSE_j(\mu_i) + MAE_j(\mu_i), \tag{31}$$

where N is the number of images in the training set, MSE is the Mean square error, and MAE is the Mean absolute error. Given the ground truth, we computed the MSE and MAE errors between the completed depth map and the ground truth of the training set and evaluated the expression (31).

For completeness, we briefly summarize the PSO algorithm, already explained in [1,26]. In each iteration of the algorithm, the candidate solutions evolve based on the dynamical equation for their position and velocity:

$$v_i^{t+1} = \omega v_i^t + \varphi_g(\mu_i^t - \mu_g) + \varphi_b(\mu_i^t - \mu_b), \tag{32}$$

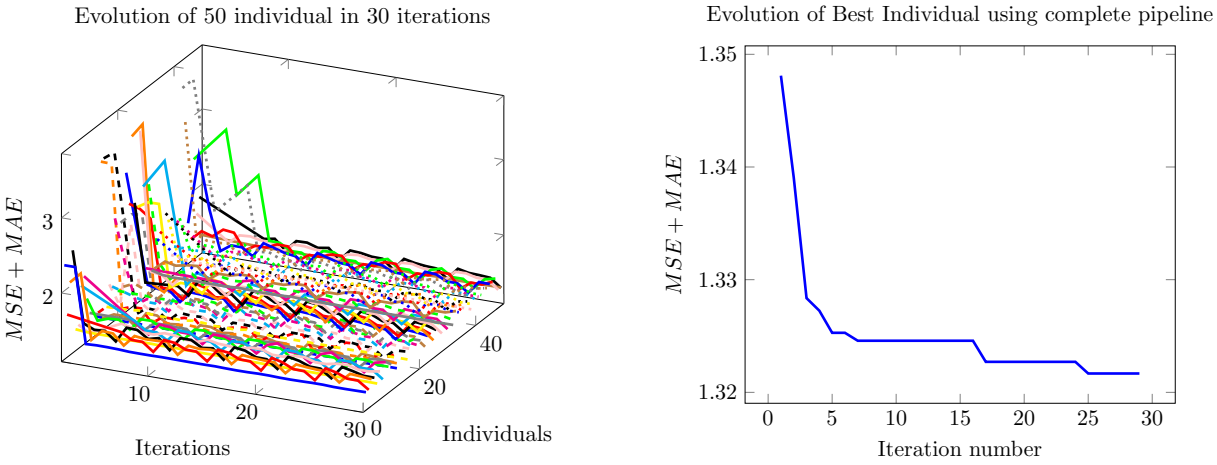
and

$$\mu_i^{t+1} = \mu_i^t + v_i^t, \tag{33}$$

where ω, μ_i, φ_g, and φ_b, $v_i \in \mathbb{R}^+$, and μ_b, μ_g represent the best current individual and μ_g the best individual of all iterations, respectively. Additionally, we used a saturation for v_i.

The training set we used to estimate model's parameters consisted of three color reference images with their corresponding incomplete depth maps and ground truth. The three reference color images selected from the KITTI depth Completion Suite [5] (see Section 4.2) to train the model are shown in Figure 3.

Figure 10a shows the evolution of the PSO algorithm. We show the fitness of 50 individuals as the iteration goes on. Figure 10b shows the fitness of the best individual in each iteration. In Figure 10b, we only observe 30 iterations, which is the stop criteria for the algorithm.



(**a**) (**b**)

Figure 10. Evolution of the PSO algorithm. Using this optimization algorithm, we selected the best set of parameters for the model. We set the number of individuals, that is, independent realizations, to 50. (**a**) Evolution of the depth error (MSE + MAE) of each of the 50 model parameters candidates and the error evolution over successive iterations. (**b**) For clarity, the evolution of the best individual from the previous 50 shown in (**a**) is shown here.

In Figure 10a, we show the evolution of each candidate μ_i in the PSO algorithm. We use 50 individuals, i.e., we have 50 curves in 30 iterations. Each individual is a possible parameter set that minimizes the $MSE + MAE$ error function, Equation (31). The evolution of each individual follows the dynamics given by Equations (32) and (33). In each iteration, we compute the performance of each individual or fitness as plotted in Figure 10a. In Figure 10b, we show the fitness of the best individual for all iterations.

The PSO algorithm was selected because most of the parameters are real numbers. We discarded other optimization methods that use discrete domains, such as the genetic algorithm. In a previous work [3], we implemented and tested the Elephant Herd Optimization (EHO) algorithm as an alternative to the PSO algorithm. The results showed that the EHO was less effective than the PSO algorithm at optimizing our objective function.

4.1. Model Training Difficulties

Several difficulties were encountered during the training process, mainly due to the initialization of the model's parameters. Some of them are detailed as follows:

1. In the training stage, we initialized the PSO with 50 instances, with each instance being a set of random values for the parameters. Some combinations of these parameters generated invalid models or, in other words, NaN (Not a number) values. CUDA and MATLAB can handle NaN values, but when MSE and MAE were computed, invalid models received a performance value of MAX_PERFORMANCE.
2. When we initialized PSO instances, for each component of the instance (or parameter), we considered a random value from the MIN_VALUE - MAX_VALUE range. This fact restricted the initial values to be positive and within reasonable ranges.
3. We limited the velocity of the PSO instances using Equation (32), using a value between -2 and 2.
4. Depth data in the data set are provided in binary format, representing an image containing a long integer. Each image was converted from long integer format to floating-point values and saved as a text file. Our code read this text file and completed the depth data. The output of the model was a floating-point image.

4.2. Experiments

We trained the model using the KITTI and the NYU_V2 data sets under the following different scenarios:

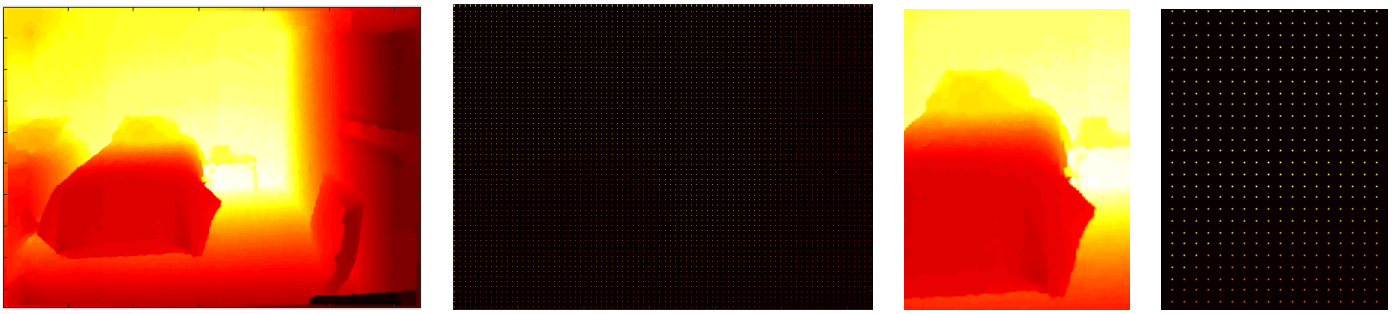
1. Using the KITTI data set considering the positive definite metric.
2. Using the KITTI data set considering the anisotropic metric.
3. Using the KITTI data set, considering the biased infinity Laplacian and the anisotropic metric.
4. Using the KITTI data set, considering the double biased infinity Laplacian and the anisotropic metric.
5. Using NYU_V2 downsampled 4× considering the positive definite metric.
6. Using NYU_V2 downsampled 4× considering the anisotropic metric.
7. Using NYU_V2 downsampled 8× considering the positive definite metric.
8. Using NYU_V2 downsampled 8× considering the anisotropic metric.
9. Using NYU_V2 downsampled 16× considering the positive definite metric.
10. Using NYU_V2 downsampled 16× considering the anisotropic metric.
11. Using the KITTI data set, we varied the number of images in the training set considering the anisotropic metric.

Considering each trained model from (**1**) to (**12**), we performed the experiments on the KITTI and NYU_V2 data sets, as summarized in Table 5.

Table 5. Experiments performed with our model in order to obtain MSE + MAE.

Experiment Number	Used Metric	Training Set	Experiment
1	positive definite Metric	KITTI training set	Interpolation of the complete KITTI data set
2	anisotropic metric	KITTI training set	Interpolation of the complete KITTI data set
3	anisotropic metric + biased infinity Laplacian	KITTI training set	Interpolation of the complete KITTI data set
4	anisotropic metric + double biased infinity Laplacian	KITTI training set	Interpolation of the complete KITTI data set
5	positive definite Metric	NYU_V2 4× training set	Upsampling of the complete NYU_V2 4× data set
6	Anisotropic Metric	NYU_V2 4× training set	Upsampling of the complete NYU_V2 4× data set
7	positive definite Metric	NYU_V2 8× training set	Upsampling of the complete NYU_V2 8× data set
8	Anisotropic Metric	NYU_V2 8× training set	Upsampling of the complete NYU_V2 8× data set
9	positive definite Metric	NYU_V2 16× training set	Upsampling of the complete NYU_V2 16× data set
10	Anisotropic Metric	NYU_V2 16× training set	Upsampling of the complete NYU_V2 16× data set
11	Anisotropic Metric	KIITI training set	Interpolation of the complete KITTI data set

Figure 11 shows an example of the subsampled depth map. In Figure 11a, we show the ground truth acquired with a Kinect sensor. In Figure 11b, we show the subsampled depth map; we took one sample every other 8×8 square pixel. Figure 11c shows a cropped area of Figure 11a and a subsampled version of Figure 11c.



(**a**) (**b**) (**c**) (**d**)

Figure 11. Example of the subsampled image. We sampled the original depth map in (**a**) every other 8×8 (or $8\times$) square pixel in (**b**). In (**c**), we zoom in on a region (on the bed) of the depth map in (**a**). In (**d**), we show a zoom of the downsampled depth map in (**c**).

Experiment number 11 considers a varying number of images in the training set. We trained the model using sets containing 3 to 18 images. In Figure 12, we show the $MSE + MAE$ error for different numbers of images.

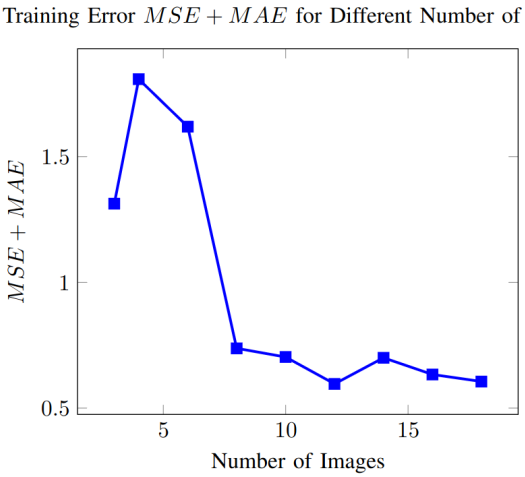


Figure 12. Error $MSE + MAE$ in the training set as a function of the number of images in the training set.

Figure 12 shows the training error as a function of the number of images. The figure shows the variability in the $MSE + MAE$ error. The fluctuation in the curve is due to the PSO algorithm's random nature and the images' natural variation. We mention that as we increased the number of images in the training set, the best-selected individual tended to require a large number of iterations for the model to converge, from 92 iterations with three images to 170 iterations with eight images, slowing down the interpolation of the depth map. Furthermore, the other candidate solutions also required a large number of iterations, contributing to the slower convergence of the PSO algorithm.

5. Results

This section summarizes the quantitative and qualitative results obtained for the KITTI and the NYU_V2 data sets.

The KITTI data set contains depth data acquired by the LiDAR sensor. Holes in depth data can be caused by the occlusion between objects in the scene, sensor misinterpretation,

transparent objects, reflections, and other effects. In Figure 13, we show examples of this lack of information, showing the reference color image and the depth ground truth.

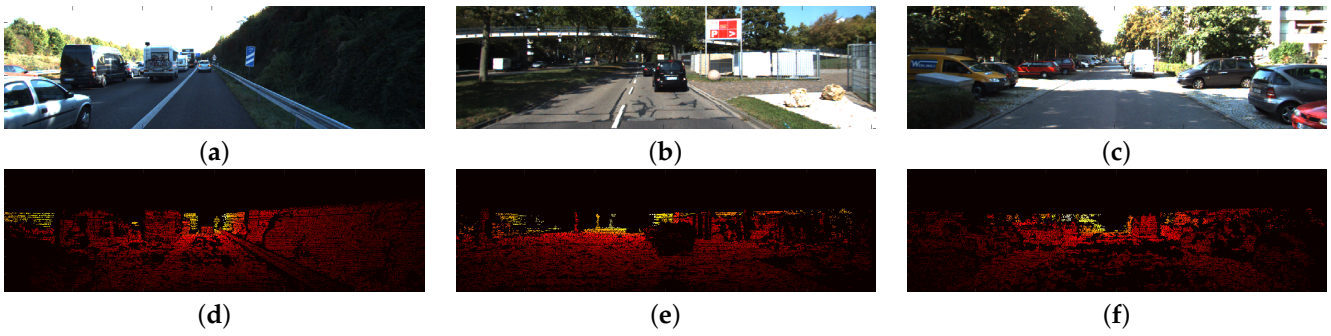


Figure 13. Examples of lack of information in depth maps. Figures (**a**–**c**) show reference color images. Figures (**d**–**f**) show depth map ground truth.

Figure 13 shows color reference images and the corresponding depth ground truth. In (a), we observe (to the left) a black van, which is highly reflective. In (d), the corresponding depth data present holes on the van surface. LiDAR failed to measure depth information on that reflective surface. Additionally, occlusion between objects created a lack of depth information below the protective railing on the right of the image. In Figure 13b,e, we observe a reflective black car and, to the right, two rocks on a white horizontal flat surface. The depth map in (Figure 13e) presents holes in the corresponding location of the black van and on the white reflective surface. In Figure 13c,f, we observe that the depth map on the yellow car (left) presents holes in its glasses due to transparent objects. Also, we observe a similar lack of information on the gray car to the right.

5.1. Results Obtained in KITTI Depth Completion Suite

We processed the complete KITTI data set and obtained the *MSE* and *MAE*. In Table 6, we show the *MSE* and *MAE* for our proposal, as well as for other models, such as Deep Fusion, Morphological Operators neural networks [23], and Convolutional Spatial Propagation Networks [24].

Table 6. Depth Completion Error in KITTI data set.

Model	MSE	MAE	MSE + MAE
Proposed model—Texture+Structure	1.1395	0.3127	1.4522
Proposed model—anisotropic metric	1.1269	0.3107	1.4376
Balanced biased infinity Laplacian	1.1254	0.3205	1.4459
double balanced biased infinity Laplacian	1.1256	0.3223	1.4479
Presented results in PDAA [1]	1.1397	0.3132	1.4529
Conv. Spatial. Prop. Networks [24]	1.0196	0.2795	1.2991
Morpho. Networks [23]	1.0455	0.3105	1.3560
Deep Fuse Networks [13]	1.2067	0.4299	1.6366

Table 6 shows the quantitative results of our proposal compared to other state-of-the-art models. Our three proposals outperform our previous work, already presented

in [1], performing better than the model considering the Texture+Structure decomposition. Comparing our results with other models, we observe that our proposals outperform Deep Fuse Networks [13] and perform similarly to Convolutional Spatial Propagation Networks [24] and Morpho Networks [23]. We emphasize that our proposals are straightforward and fast to implement. Completing $u(\mathbf{x})$ involves a simple weighted average and an iterative procedure.

Table 6 shows that the proposed anisotropic metric model is our best proposal considering MSE+MAE. It is noteworthy that the model presenting the best performance is the simplest one, i.e., infinity Laplacian and the anisotropic metric.

In Figure 14, we show qualitative results obtained by our models in the KITTI data set. Figure 14e,f show the minimum MSE obtained by our model in the KITTI data set, with an error of 53.78 cm and 52.09 cm, respectively.

Figure 14. Examples of qualitative and quantitative results obtained by our best model in the KITTI data set. We show color reference images in (**a**,**b**). In (**c**,**d**), we show the sparse depth map used as input to our model. In (**e**,**f**), we show the completed depth map and the *MSE* obtained in each image.

In Figure 15, we show qualitative results where our model fails to complete the task. In Figure 15g–i, we observe that our model cannot recover the car geometry due to transparent objects or excessive reflections. In these three images, we obtained the most significant *MSE* errors. In (g) we obtained $MSE = 3.2025$, in Figure 15h we obtained $MSE = 3.1126$, and in Figure 15i we obtained $MSE = 2.7428$.

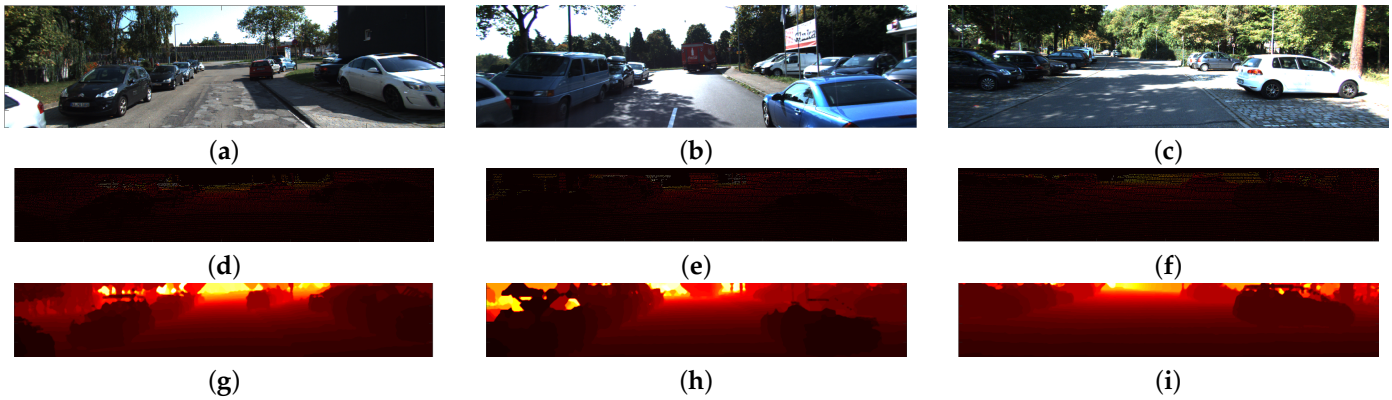


Figure 15. Examples of qualitative and quantitative errors obtained by our model in the KITTI data set. We show our worst results in the KITTI data set. In (**a**–**c**), we show color reference images. We present the sparse depth map used as input to our model in (**d**–**f**). In (**g**–**i**), we show the completed depth map.

5.2. Experiments Varying the Number of Images in the Training Set

Experiment 11 involved varying the number of images in the training set. We trained the model and then we evaluated the model using the complete KITTI data set.

In Table 7, we show the experiment's results.

Table 7. Depth Completion Error obtained varying the number of images in the training set.

Number of Images in the Training Set	MSE cm	MAE cm	Error $MSE + MAE$ cm
3	1.1254	0.3205	1.4459
4	1.1272	0.3139	1.4411
6	1.2617	0.4237	1.6854
8	1.1355	0.3226	1.4581
10	1.2826	0.4247	1.7073
12	1.1353	0.3234	1.4586
14	1.4396	0.7735	2.2132
16	1.4412	0.7675	2.2087
18	1.4365	0.7603	2.1968

In Figure 16, we simultaneously plot the obtained results in Figure 12 and Table 7. We show the results in the training set and in the validation set.

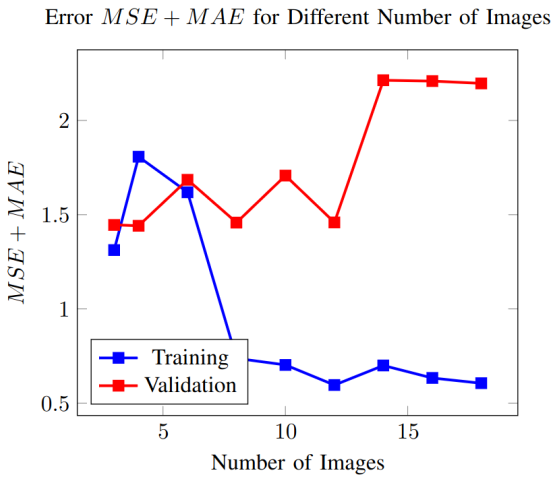


Figure 16. Error $MSE + MAE$ in the training and validation set as a function of the number of images in the training set.

In Figure 16, we observe that as the number of training images increases, the training error decreases and finally saturates (curve in blue). However, the error in the validation set tends to increase as the number of images grows (curve in red). This behavior shows the model is overfitting to the training set. To obtain better results with a large set of training images, the PSO algorithm may overfit the model, i.e., it can reach $MSE + MAE = 0.7029$ with ten images.

Considering three or four images in the training set, we had higher training errors; however, we obtained the minimum error in the validation set.

This study highlights the trade-off between good performance in the training set and good performance in the validation set. Therefore, our decision to take three images in the training set is supported by the fact that it leads to the best performance in the validation set and suggests more generalization capabilities.

5.3. Results Obtained in NYU _V2

Table 8 shows our results for different subsampling factors in the upsampling task. We compare the results obtained by our proposal with different models available in the literature regarding the completion error (*MSE*).

Table 8. Depth Completion Error obtained by different models in NYU_V2.

Model	Error 4× cm	Error 8× cm	Error 16× cm
Proposed model	8.31	9.38	17.56
Proposed model—anisotropic metric	8.31	9.35	17.16
Bicubic	8.46	14.22	22.32
TGV [10]	6.98	11.23	28.13
Ham [28]	5.27	12.31	19.24
JBU	4.07	8.39	13.25
Park [2]	5.21	9.56	18.10
DJF [29]	3.54	6.20	10.21

We plot the data in Table 8 in Figure 17.

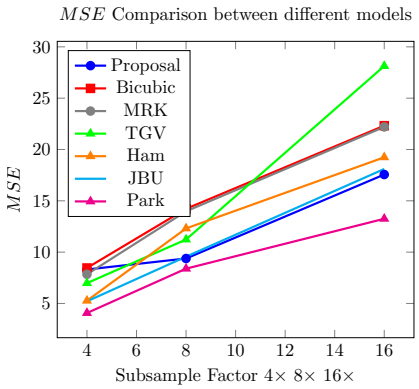


Figure 17. Comparison of obtained *MSE* between models: Our proposal, Bicubic, TGV [10], Ham [28], JBU, and Park [2].

In Figure 17, we observe that our proposal (in blue) outperforms the model Total Generalized Variation [10], Filtering using static and dynamic guidance [28], and Bicubic interpolation for upsampling factors of 8× and 16×. The JBU and Ham models [28] perform better than ours.

In Figure 18, we show quantitative and qualitative results obtained by the proposal on the images already presented above in Figure 9. Depth maps were downsampled by a factor of eight.

Figure 18 shows color frames and their corresponding interpolated depth maps. Figure 18g shows the completed depth map of the downsampled depth map. Similar results are presented in Figure 18h,i.

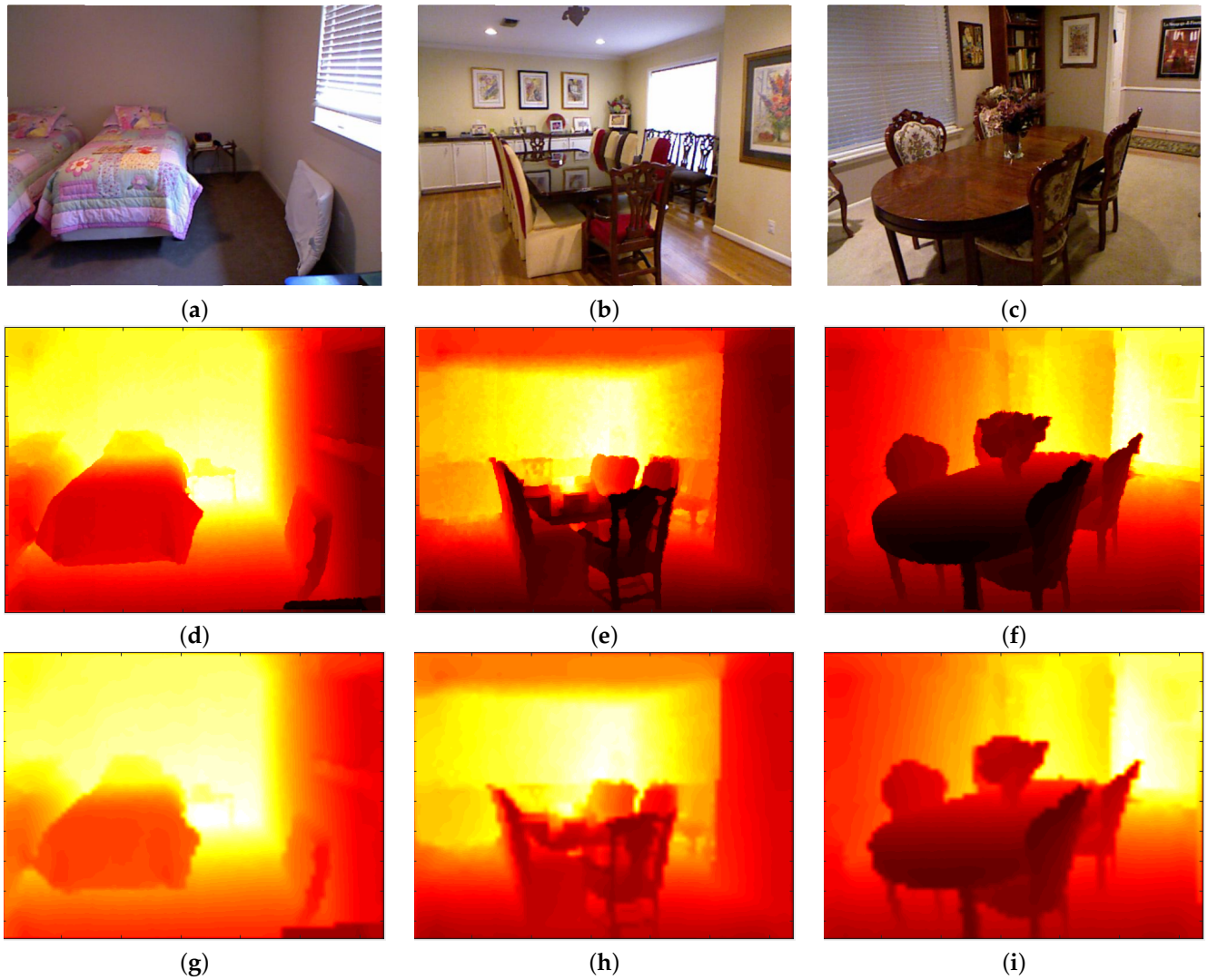


Figure 18. Examples of completed depth maps in the NYU_V2 data set. Figures (**a**–**c**) show examples of three color reference images, and (**d**–**f**) show their corresponding ground truth, respectively. In (**g**), we obtained an $MSE = 3.94$, in (**h**) we obtained $MSE = 10.69$, and in (**i**) we obtained $MSE = 10.90$. In these figures, we show the completed depth maps with an upsampling factor of $8\times$.

5.4. Ablation Test

We performed an ablation test on our proposed model. In this ablation test, we used the NYU_V2 data set with a factor $4\times$. In each evaluation, we disabled one of the stages of the model while keeping the other stages active. In each test, we turned off the convolution stage (CS1) and the convolutional stage (CS2). We show the results of the ablation test in Table 9.

Table 9. Results of the ablation test.

Stage	MSE
Complete	8.3136
Convolutional Stage CS1	8.3272
Anisotropic Metric	8.3140
Convolutional Stage CS2	8.6529

In Table 9, we show the results of the ablation test. By turning off the convolutional stage CS1, we observe that the MSE increases by 0.0146 cm; by changing the anisotropic metric for the previous metric, the MSE increases by 0.0004 cm. Turning off the convolu-

tional stage CS2, the MSE error increases by 0.3393 cm. These increments show that given the data set NYU_V2 and the estimated parameters, the most critical stage of the proposal is the final stage, which eliminates outliers and noise.

5.5. Discussion

In summary, we present the results obtained by our proposal in the KITTI data set and in the NYU_V2 data set in Tables 10 and 11, respectively.

Table 10. Best results obtained in KITTI data set.

Model	MSE	MAE
Anisotropic Metric	1.1269	0.3107

Table 11. Best results obtained in NYU_V2 data.

Model	MSE ×4	MSE ×8	MSE ×16
Anisotropic Metric	8.31	9.38	17.56

In Table 10, in the second row, we show the results obtained by our proposal in the KITTI data set. We obtained an $MSE = 1.1269$ in the KITTI data set, outperforming our previous published version of the model in [1] ($MSE = 1.1397$) and other similar contemporaneous models [13]. In Table 11, in the NYU_V2 data set, we obtained MSEs of 8.31, 9.35, and 17.56 with an upsampling factor of 4×, 8×, and 16×, respectively. Our results outperform many contemporaneous models, such as [28] and [10].

Last but not least, we would like to mention the main limitations of our model and our evaluation:

1. The model struggled to accurately fill in objects with weak edges in the reference image. In the case of objects with weak edges, the model failed to accurately propagate the depth information correctly, causing data from different objects to merge. A possible solution to tackle this problem could be to use a semantic segmentation of the reference color image and use superpixel boundaries in order to stop diffusion.
2. The model faced challenges when dealing with transparent and reflective objects. We will tackle this problem in our future work.
3. Performance was evaluated using only two data sets. We consider that having only two data sets is insufficient to generalize our conclusions. Therefore, we plan to extend the evaluation to more data sets. In any case, the preliminary results are promising, particularly in using this technique in real-world scenarios and applications such as 3D reconstruction of an urban scene. An example of the depth completion for an urban scene is shown in Figure 19.

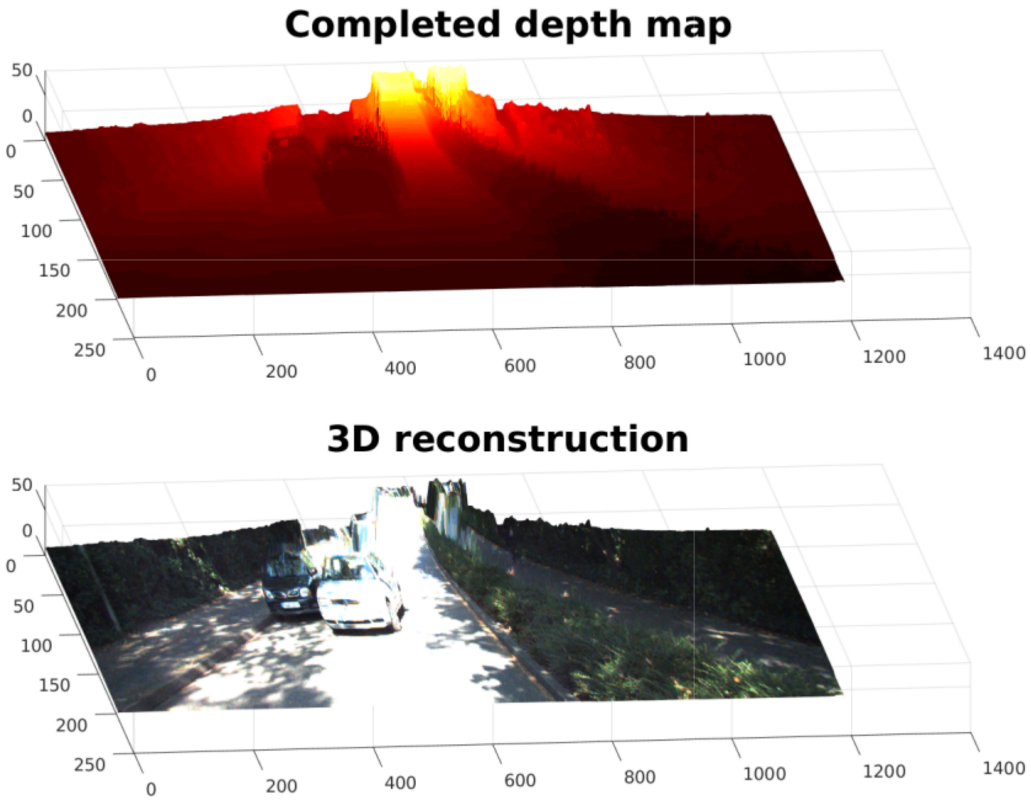


Figure 19. A 3D reconstruction of an urban scene from a depth map and a color frame. The first row shows the 3D interpolated depth of an urban scene. The second row shows the reconstruction of the 3D scene created using the color reference and the completed depth map.

6. Conclusions

This paper extends the work presented in [1], evaluating new variations of the original model to interpolate incomplete depth maps. The results have been qualitatively and quantitatively evaluated in two data sets, KITTI and NYU_V2.

The main conclusions of this work are as follows: (i) the use of an anisotropic metric improves model performance; (ii) the use of a Texture+Structure decomposition as a pre-filtering stage did not have a significant impact; and (iii) the ablation test showed that the model's final stage is an essential component in the pipeline, with its suppression leading to an MSE increase of approximately 4%. We also show that our model outperforms state-of-the-art alternatives with similar levels of complexity.

The model shows a performance superior to or at least similar to other state-of-the-art algorithms that are not based on deep neural networks. Nevertheless, its complexity and execution were significantly faster than those approaches, outperforming most of them by an order of magnitude. In the context of the current evaluation, our approach outperformed TGV [10] and JBU in the NYU_V2 data set, and is comparable to Morpho Networks [23] and Convolutional Spatial Networks [24].

We varied the number of images included in the training set. The results show that as the number of considered images grows, the PSO algorithm tends to overfit the model to obtain better results in the training set. This fact comes at the expense of compromising the model's performance in the validation test. The model achieves acceptable performance when using only three images for training. For training sets using more than three images, the PSO algorithm selects the best parameters, which present a large number of iterations, increasing the complexity of the model, making the execution of the model run slowly and be more demanding on computational resources.

A hybrid approach is advantageous as it combines the generalization capabilities of classical models with the high precision provided by new features obtained through

convolutional stages. Also, the flexible structure of the pipeline lets the model reach its maximum performance considering the used interpolation model.

Experimentally, we evaluated the model only in two data sets: KITTI and NYU_V2. In order to assess the generalization capabilities of the model, it is necessary to extend the evaluation to other contemporary data sets, such as Middlebury or Virtual KITTI. Furthermore, the parameter estimator (PSO) has additional parameters that require optimization. We think it is possible to obtain better performance by optimizing the parameters of the PSO algorithm. We propose investigating nested optimization to jointly estimate both the parameters of the model and those of the PSO algorithm.

Future work will consider the implementation of a better approximation of the geodesic distance, evaluating the model in other data sets (synthetic and real data sets), and matching patches instead of matching individual pixels. When we determine the $u(\mathbf{y})$ that maximizes the positive eikonal operator and the $u(\mathbf{z})$ that minimizes the negative eikonal operator, we compare pixels; more robust matching could be performed using patches instead of matching pixels. This fact will increase the confidence of the matching.

We will also improve the model's result in areas of the color image with weak boundaries and in the presence of transparency and intense reflections.

Author Contributions: The development of this work, F.C. and V.L.; contribution to conceptualization, V.L. and F.C.; methodology, V.L. and F.C.; software, V.L.; validation, V.L.; analysis, F.C.; bibliography and references, V.L. and F.C.; writing—original draft preparation, V.L. and F.C.; review and editing, F.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available in The KITTI Vision Bench-mark Suite at https://www.cvlibs.net/datasets/kitti/eval_depth.php?benchmark=depth_completion and NUYDepthDatasetV2 at https://cs.nyu.edu/~fergus/datasets/nyu_depth_v2.html, accessed on 23 April 2024.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Lazcano, V.; Calderero, F. Hybrid Model Convolutional Stage-Positive Definite Metric Operator-Infinity Laplacian Applied to Depth Completion. In Proceedings of the IEEE 2022 Tenth International Symposium on Computing and Networking Workshops (CANDARW), Himeji, Japan, 21–24 November 2022; pp. 174–179. [CrossRef]
2. Park, J.; Kim, H.; Yu-Wing, T.; Brown, M.; Kweon, I. High quality depth map upsampling for 3D-TOF cameras. In Proceedings of the 2011 International Conference on Computer Vision, IEEE, Barcelona, Spain, 6–13 November 2011; pp. 1623–1630. [CrossRef]
3. Lazcano, V.; Calderero, F.; Ballester, C. Comparing Different Metrics on an Anisotropic Depth Completion Model. *Int. J. Hydrid Intell. Syst.* **2021**, *17*, 87–99. [CrossRef]
4. Lu, K.; Barnes, N.; Anwar, S.; Zheng, L. From Depth What Can You See? Depth Completion via Auxiliary Image Reconstruction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Seattle, WA, USA, 13–19 June 2020.
5. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets Robotics: The KITTI Dataset. *Int. J. Robot. Res. (Ijrr)* **2013**, *32*, 1231–1237. [CrossRef]
6. Li, Y.; Huang, J.B.; Ahuja, N.; Yang, M.H. Joint Image Filtering with Depth Convolutional Networks. *Trans. Pattern Anal. Mach. Intell.* **2019**, *41*, 1909–1923. [CrossRef] [PubMed]
7. Lai, W.S.; Huang, J.B.; Ahuja, N.; Yang, M.H. Fast and accurate Image Super-Resolution with Deep Laplacian Pyramid Network. *Trans. Pattern Anal. Mach. Intell.* **2019**, *41*, 2599–2613. [CrossRef] [PubMed]
8. Imran, S.; Long, Y.; Liu, X.; Morris, D. Depth Coefficients for Depth Completion. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 12438–12447.
9. Zhang, Y.; Funkhouser, T. Deep depth completion of a single rgb-d image. In Proceedings of the IEEE Internatinal Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, 18–22 June 2018.
10. Ferstl, D.; Reinbacher, C.; Ranftl, R.; Ruether, M.; Bischof, H. Image Guided Depth Upsampling Using Anisotropic Total Generalized Variation. In Proceedings of the 2013 IEEE International Conference on Computer Vision, Sydney, NSW, Australia, 1–8 December 2013; pp. 993–1000. [CrossRef]

11. Xu, Y.; Zhu, X.; Shi, J.; Zhang, G. Depth completion from sparse lidar data with depth-normal constraints. In Proceedings of the International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019.
12. Park, J.; Joo, K.; Hu, Z.; Liu, C.; Kweon, I.S. Non-local spatial propagation network for depth completion. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020.
13. Shivakumar, S.; Nguyen, T.; Miller, I.; Chen, S.; Kumar, V.; Taylor, C. DFuseNet: Deep Fusion of RGB and Sparse Depth Information for Image Guided Dense Depth Completion. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 27–30 October 2019; pp. 13–20. [CrossRef]
14. Hu, M.; Wang, S.; Li, B.; Ning, S.; Fan, L. Penet: Towards precise and efficient image guided depth completion. In Proceedings of the International Conference on Robotics and Automation, Xi'an China, 30 May–5 June 2021.
15. Lin, Y.; Cheng, T.; Zhong, Q.; Zhou, W.; Yang, H. Dynamic Spatial Propagation Network for Depth Completion. In Proceedings of the AAAI 2022 Conference on Artificial Intelligence, Virtual, 22 February–1 March 2022.
16. Tang, J.; Tian, F.; Feng, W.; Li, J.; Tan, P. Learning guided convolutional network for depth completion. *IEEE Trans. Image Process.* **2021**, *30*, 1116–1129 . [CrossRef] [PubMed]
17. Bai, L.; Zhao, Y.; Elhousni, M.; Huang, X. DepthNet: Real-Time LiDAR Point Cloud Depth Completion for Autonomous Vehicles. *IEEE Access* **2020**, *8*, 227825–227833. [CrossRef]
18. Caselles, V.; Igual, L.; Sander, O. An axiomatic approach to scalar data interpolation on surfaces. *Numer. Math.* **2006**, *102*, 383–411. [CrossRef]
19. Lazcano, V.; Calderero, F.; Ballester, C. Depth Image Completion Using Anisotropic Operators. In *Proceedings of the 12th International Conference on Soft Computing and Pattern Recognition (SoCPaR 2020). Advances in Intelligent Systems and Computing*; Springer: Cham, Switzerland , 2020.
20. Krauss, B.; Schroeder, G.; Gustke, M.; Hussein, A. Deterministic Guided LiDAR Depth Map Completion. In Proceedings of the IEEE Intelligent Vehicles Symposium (IV), Nagoya, Japan, 11–17 July 2021; pp. 824–831. [CrossRef]
21. Saidi, T.; Kohuas, A.; Amira, A. Implementation of a real-time stereo vision algorithm on a cost-effective heterogeneous multicore platform. *Concurr. Comput. Pract. Exp.* **2023**, *35*, 1–22. [CrossRef]
22. Eldesokey, A.; Felsberg, M.; Holmquist, K.; Persson, M. Uncertainty-Aware CNNs for Depth Completion: Uncertainty from Beginning to End. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020.
23. Dimitrievski, M.D.; Veelaert, P.; Philips, W. Learning Morphological Operators for Depth Completion. In Proceedings of the Advanced Concepts for Intelligent Vision Systems Conference, IEEE, Poitiers, France, 24–27 September 2018.
24. Cheng, X.; Wang, P.; Yang, R. Learning Depth with Convolutional Spatial Propagation Network. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 2361–2379. [CrossRef] [PubMed]
25. Silberman, N.; Hoiem, D.; Kohli, P.; Fergus, R. Indoor Segmentation and Support Inference from RGBD Images. In Proceedings of the ECCV, Firenze, Italy, 7–13 October 2012.
26. Lazcano, V.; Ramírez, I.; Calderero, F. Anisotropic Operator Based on Adaptable Metric-Convolution Stage-Depth Filtering Applied to Depth Completion. In *Pattern Recognition*; Lu, H., Blumenstein, M., Cho, S.B., Liu, C.L., Yagi, Y., Kamiya, T., Eds.; Springer: Cham, Switzerland, 2023; pp. 1–14.
27. Lazcano, V.; Calderero, F. A Discussion on Variants of an Anisotropic Model Applied to Depth Completion. In *Advanced Research in Technologies, Information, Innovation and Sustainability*; Guarda, T., Portela, F., Diaz-Nafria, J.M., Eds.; Springer: Cham, Switzerland, 2024; pp. 3–16.
28. Ham, B.; Cho, M.; Ponce, J. Robust image filtering using joint static and dynamic guidance. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015.
29. Li, Y.; Huang, J.; Ahuja, N.; Yang, M. Deep Joint Image Filtering. In *Computer Vision—ECCV 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer: Cham, Switzerland, 2016; pp. 154–169.