Self Determining Structure Model for Depth Map Completion

Vanel Lazcano[1,*], Anthony D. Cho[1], and Felipe Calderero[2]

[1]Escuela de Ingeniería, Facultad de Ciencias, Ingeniería y Tecnología, Universidad Mayor, Santiago 7500994, Chile
[2]Data and AI Department, Nucleo Digital School, Madrid, Spain
Email: vanel.lazcano@umayor.cl (V.L.); anthony.cho@umayor.cl (A.D.C.); felipecalderero@gmail.com (F.C.)
*Corresponding author

Manuscript received January 7, 2025; revised March 18, 2025; accepted November 7, 2025; published May 15, 2026

***Abstract*—In this paper, we present a comprehensive approach for depth map completion using a variable pipeline model. The pipeline is designed to be flexible, and its structure is dynamically determined during the training phase by minimizing a fitness function. To achieve this objective, we used the Particle Swarm Optimization (PSO) technique, which efficiently explores the solution space to optimize key parameters. The pipeline itself consists of three main stages: a convolutional stage (Lab Convolution), an Interpolation Model (IM), and a post-Convolutional Stage (SC2). During training, our model automatically adjusts several aspects, including the selection of filter parameters, interpolator settings, and the number of filters used at each stage. Moreover, the pipeline is capable of interchanging the execution order between the stages, either processing in the sequence of Lab Convolution-IM-SC2 or SC2-IM-Lab Convolution. This flexible structure allows for the discovery of better configurations that significantly improve the performance of depth map completion tasks. Our approach demonstrates superior results by adapting to the specific characteristics of the input data.**

***Keywords*—variable pipeline, particle swarm optimization, depth map completion**

I. Introduction

Classical models based on energy minimization suffer criticism, coming from the machine learning community, that these kinds of models are designed by hand [1]. What the machine learning community forgot is that neural network architectures are also designed by hand.

In this work, we present an in-depth model that considers a convolutional stage and an interpolator applied to depth map completion. Our proposal estimates its structure, parameter filters, number of filters required, number of iterations, and other values in the training stage.

We complete a sparse depth map of an urban scene using standard datasets [2, 3] that contain a sparse depth map and its corresponding color reference image. The depth information was obtained by a LiDAR sensor, and the corresponding color image by a RGB camera.

Among the related works, reconfigurable systems such as neural networks have been addressed using different approaches: an adaptive boost to iteratively construct the structure of a neural network, adaptive kernels, genetic algorithms to define the architecture of a neural network, and many others.

In Ref. [4], a variable Convolutional Neural Network (CNN) structure was introduced. The authors allow the models to have variable size and shape by skipping links between nodes. They used a genetic algorithm to estimate the network structure, with the goal of enhancing model performance. Their results were evaluated across three datasets, demonstrating an improvement of approximately 40% in classification tasks.

Zamora-Esquivel *et al.* [5] presented adaptive kernels applied to image processing. The authors developed a strategy to compute parameters for a dynamically changing kernel in a convolutional layer from an image. These adaptive kernels facilitate accurate recognition while requiring significantly less memory, employing fewer kernels and layers compared to traditional Convolutional Neural Network (CNN) configurations. They applied their model to a classification task using the Modified National Institute of Standards and Technology (MNIST) dataset.

Cortes *et al.* [6] showed an adaptively learning neural network, namely ADANet. This model simultaneously estimates the network structure and its weights. The model starts with a very simple structure and gradually increases the units' number and hidden layers using their methodology and balancing the model complexity with an empirical risk minimization. Their method consists of creating sub-networks guided by their learning algorithm in order to connect the most relevant ones, and was applied to binary classification problems showing competitive results.

Some recent work focuses on the application of anisotropic cost in combination with uncertainty quantification on missing pixels [7], anisotropic metric to improve interpolation to fill holes on a depth map [8], accelerated diffusion model in conjunction with Markov chain to control the transition between the high and low quality images by shifting their residuals [9], and others [10, 11].

The rest of this work is organized as follows. Section II we describe our model, followed by Section III presenting the performed experiments. And finally in Sections IV and V, our results and conclusions.

II. Model

In this section, we provide a detailed explanation of our proposal regarding the pipeline structure illustrated in Fig. 1. Depending on a binary variable (*Reversible*) the pipeline can be carried out following the light blue path or the light orange path.

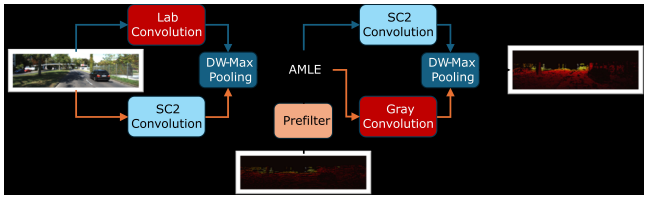


Fig. 1. Structure of the variable architecture model.

The main idea behind this model is to take as input an RGB color reference image and a sparse depth map to complete the depth information via interpolation. Prior to undergoing the

method, an RGB image is converted into Lab color space according to Ref. [12], since it works better in our proposal compared to other color spaces.

A. Light Blue Path

Taking into account the light blue path shown in Fig. 1, each Lab component of the image is convolved with a set of Gabor filters (red block) to enforce edges. Here, we considered *NFilt* Gabor filters, and each Gabor filter has its own parameters (σ and ω), such parameters define the spatial support of each filter and the spatial frequency.

Continuing with the light blue path, we applied a Depth-Wise Max-Pooling to obtain a color feature image. The Absolutely Minimizing Lipschitz Extension (AMLE) interpolator takes as a second input the sparse depth map. Following the black path, Fig. 1 shows a prefilter (a light red block) applied to the sparse depth map, this prefilter eliminates outliers and noise from the depth data. The prefiltered depth data and the color feature image are the inputs to the interpolator, which uses the infinity Laplacian to interpolate the available depth information and is guided by the color feature image. The interpolated data is the solution of the infinity Laplacian given the sparse depth information. Basically, the interpolation method performs an anisotropic diffusion guided by the geometry of the scene provided by the color feature image.

Proceeding along the light blue path, the interpolated depth data is filtered using a number of (*NFilt2*) weighted masks. Each weighted mask is a 3×3 grid with nine parameters to estimate. Finally, a Depth-Wise Max-Pooling is applied to obtain the filtered interpolated depth data.

We remark that the number of filters (*NFilt* and *NFilt2*), parameters of the filters, parameters of the interpolator, and the binary parameter *Reversible* are estimated in the training stage.

B. Light Orange Path

On the other hand, following the light orange path illustrated in Fig. 1, each Lab-component of the color image is convolved with a weighted mask located in the SC2 convolution stage (light blue block), which helps to enforce edges for the anisotropic diffusion that performs the interpolator. The output of this convolutional stage is submitted to Depth-Wise Max-Pooling to obtain a usable color feature image for the infinity Laplacian method.

As we have explained in Section II-A, the available sparse depth information is filtered and used as input for the infinity Laplacian method jointly with the color feature image. The solution of the infinity Laplacian is convolved with a battery of Gabor filters to eliminate noise and outliers. And finally, the completed depth data is obtained through Depth-Wise Max-Pooling.

C. Lab Convolution

This stage consists of *NFilt* Gabor filters ranging from 1 to 8, and the number of Gabor filters *NFilt* is determined in the training stage. Fig. 2 shows an example of the output of the Lab Convolution.

As shown in Fig. 2, the output of this process is a color image that high-lights enforced edges. This output shows that the horizontal edges are enforced. This phenomenon can be attributed to the LiDAR sensor, which scans the urban environment horizontally, resulting in data that is oriented horizontally. To enhance the depth information, it is essential to diffuse the available data in the vertical direction. The strong edges in the color feature image inhibit diffusion in that direction, thereby preserving the geometry of the scene.

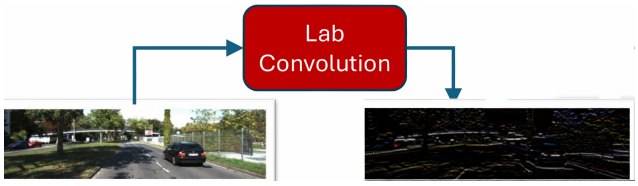


Fig. 2. Example of the output of this convolutional stage and the depth-wise max-pooling.

Fig. 3 shows a 3D representation of the Gabor filter using the selected parameters. This filter is normalized to 1.

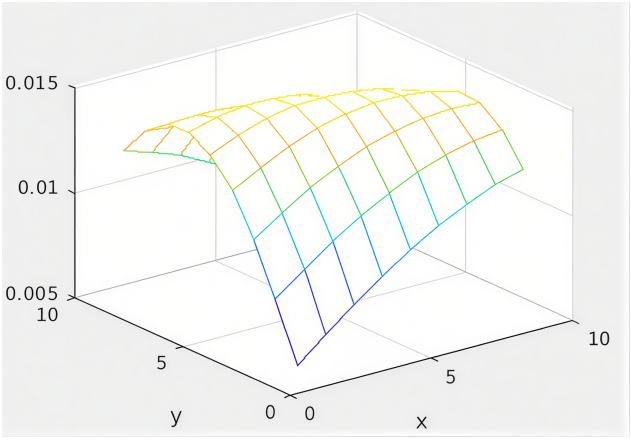


Fig. 3. Gabor filter with selected parameters.

D. Interpolator

We use the infinity Laplacian to interpolate the data. As outlined in Ref. [13], the infinity Laplacian is governed by several key axioms: i) Boundary Value, ii) Comparison Principle, iii) Stability Principle, iv) Regularity Principle, v) Gray Scale Shift Invariance, and vi) Linear Gray Scale Invariance. These axioms not only facilitate the implementation of the infinity Laplacian but also provide a straightforward solution for several computer vision challenges, including applications in embedded systems.

The solution of the infinity Laplacian is a simple weighted sum of the data around a point x in the image domain. Let us define Ω as a rectangular domain $\Omega \subset \mathbb{R}^2$, u the interpolated depth data u: $\Omega \rightarrow \mathbb{R}$, with $u(\partial\Omega) = \theta$ as the sparse depth map. Lets consider $N(x)$ a neighborhood (of size = *radius*) around x, and $y \in N(x)$ be a point that maximizes $\frac{u(y)-u(x)}{d_{xy}}$ and $z \in N(x)$ be a point that minimizes $\frac{u(y)-u(x)}{d_{xy}}$, where d_{xy} and d_{xz} is the distance between x and y, and x and z, respectively. Given this definition the infinity Laplacian is given by the following Eq. (1),

$$u^{n+1}(x) = \frac{u^n(y)d_{xz}+u^n(z)d_{xy}}{d_{xy}+d_{xz}} \quad (1)$$

which is the iterative version of the infinity Laplacian [14] for $n = 0, 1, 2, \ldots$.

The distance d_{xy} is defines as:

$$d_{xy} = k_x \| x - y \|^2 + k_c \| I(x) - I(y) \|^2 \quad (2)$$

where k_x, $k_c \in \mathbb{R}$ are estimated in the training stage.

Eq. (2) shows that the metric is composed of two terms: the first term is the spatial distance and the second term is the photometric distance considering color image intensities.

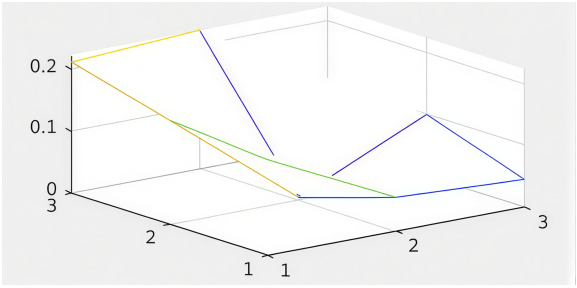
E. SC2 Convolution

This stage is composed of 3×3 size filters. The number of filters, *NFilt2*, considered in this stage is estimated in the training stage.

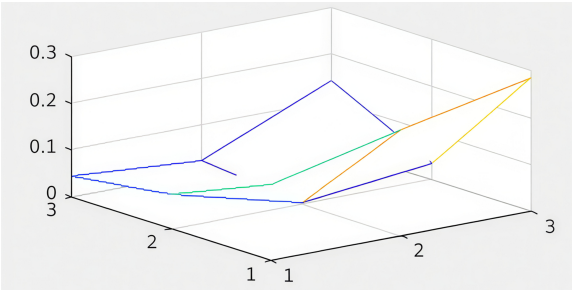
Fig. 4 shows a filter structure applied to interpolated depth data to eliminate outliers and noise. Fig. 5 shows the 3D plot of selected filters in the training stage, which were normalized to 1. In this case, the number of selected filters estimated was *NFilt2* = 2.

$w_{1,1}^k$	$w_{1,2}^k$	$w_{1,3}^k$
$w_{2,1}^k$	$w_{2,2}^k$	$w_{2,3}^k$
$w_{3,1}^k$	$w_{3,2}^k$	$w_{3,3}^k$

Fig. 4. Weight mask applied in SC2 convolutional stage where each w_{ij}^k is a weight of the mask, and k = 1, ..., *NFilt2*.



(a)

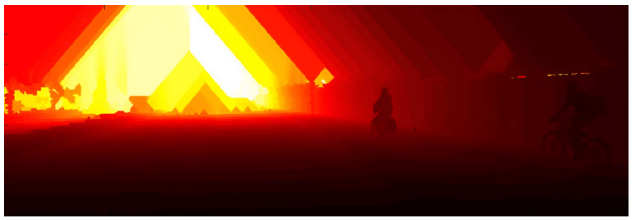


(b)

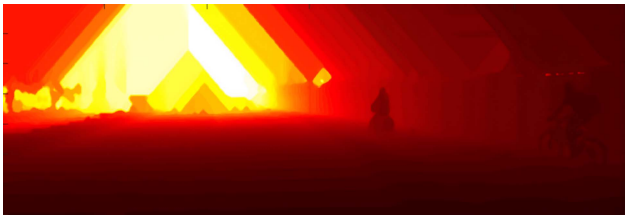
Fig. 5. Selected average filters: (a) first selected filter shape; (b) second selected filter shape.

The reason why we use this stage as post-filtering after the infinity Laplacian is to smooth the interpolated depth map and remove outliers. Fig. 6 shows an example of depth map before and after post-filtering.

Fig. 6(a) shows the interpolated data before applying post-filtering, this depth map provides a Mean Square Error (MSE) of 0.9972, which is higher than the resulting one after post-filtering, which provides an MSE = 0.9207. Thanks to the filtering, we can observe in Fig. 6(b) that the largest values of the estimated data (outlier) have been removed (this can be appreciated in the area just above the person riding a bicycle).



(a)



(b)

Fig. 6. Examples of interpolated depth data with and without post-filtering. (a) MSE = 0.9972; (b) MSE = 0.9207.

F. Implementation

Our model was implemented in C++/CUDA in conjunction with the PSO method in MATLAB, and it was run in a system equipped with an Intel i7-11800H processor (3.30GHz), 16 GB of RAM, NVIDIA RTX 3070, and Ubuntu 22.04 operating system.

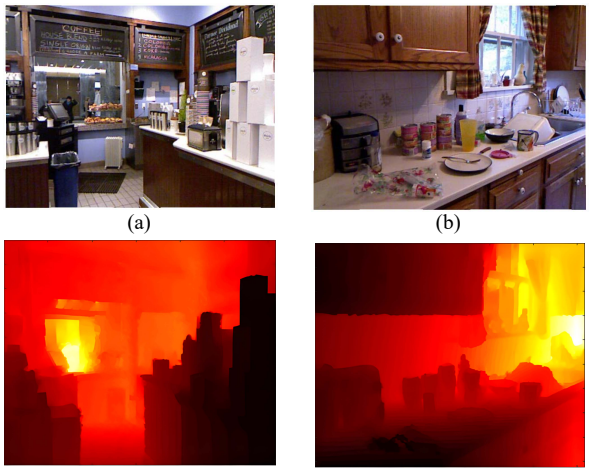
III. Dataset and Training

In order to train and assess the model, we have considered the KITTI dataset [2] and the NYU-Depth V2 dataset [3]. The KITTI dataset comprises 1000 color images along with their corresponding depth maps, which were acquired using a LiDAR sensor in an urban environment. Additionally, this dataset includes a high-density depth map for each sparse depth map, allowing us to evaluate the performance of depth completion models effectively. Fig. 7 shows the reference color image used to train the model.

Fig. 7. Four reference color images to train the model.

The training images were randomly selected from the set of images in the dataset and the model was trained only once with this training dataset.

The NYU-Depth V2 dataset is composed of 1449 indoor color reference images and their respective depth map acquired by a Microsoft Kinect sensor. For our analysis, we randomly selected four color images from the first 1000 images, while the remaining images were utilized to evaluate the performance of the model. Fig. 8 shows two examples of color reference images and their respective depth data from the NYU-Depth V2 dataset.



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

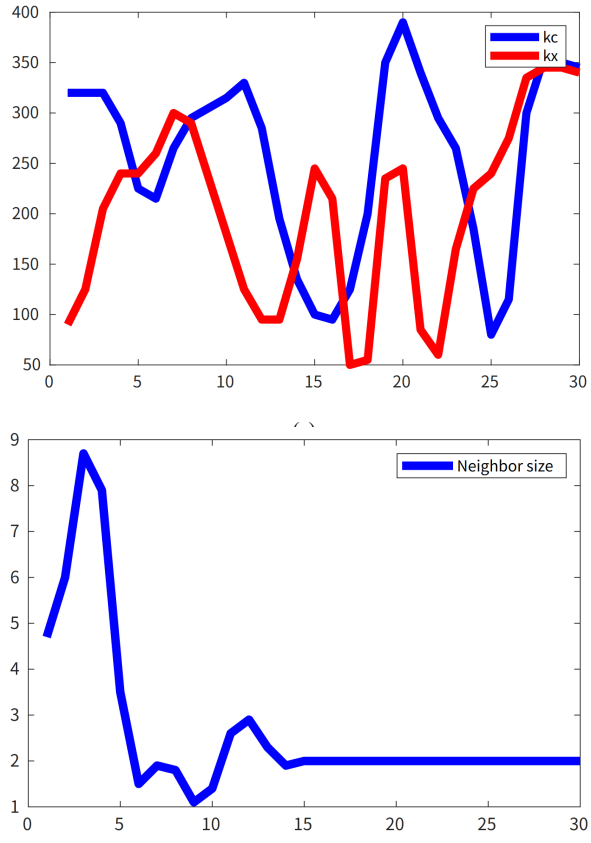
Fig. 8. Figure shows two color reference images along with their corresponding depth map. The depth map is color coded using MATLAB colormap jet, light yellow represents depth large values and dark red represents depth small values: (a) and (b) are two image references from the NYU-Depth V2 dataset, and (c) and (d) their ground truth depth maps, respectively.

A. Training the Model

Four images extracted from the KITTI dataset were used to train the model, and its parameters are estimated via the Particle Swarm Optimization (PSO) algorithm by minimizing the sum of MSE and Mean Absolute Error (MAE) of the obtained depth data.

Fig. 9(a) shows the evolution of the parameters k_x and k_c defined in Eq. (2), where we can observe that both values converge to 350 approx. Fig. 9(b) shows the evolution of the radius given the size of the neighborhood, indicating that the radius which minimizes the *MSE* + *MAE* is *radius* = 2. This implies that the model considers a neighborhood of ((2 × *radius* + 1) × (2 × *radius* + 1)), which corresponds to a neighborhood of 5×5 pixels. Starting with *radius* = 9 evolving to *radius* = 1 for *iteration* = 8, and finally converges to *radius* = 2.

Fig. 10(a) shows the evolution of the *Reversible* parameter. Initially, the *Reversible* parameter is set around 1 (larger than 0.5), indicating that the computing sequence follows Lab Convolution-AMLE-SC2. After two iterations, the *Reversible* reaches 0.4, which is less than 0.5, indicating a switch to the SC2-AMLE-Lab Convolution sequence. In Fig. 10(b), we can see the learning curve or error (*MSE* + *MAE*) of the best individual in each iteration. It is evident that performance evolves in correlation with the value of *Reversible* parameter. When the *Reversible* value falls below 0.5, the error decreases from 1.9 m to 1.84 m.



(b)

Fig. 9. Evolution of parameters' estimation in 30 iterations. (a) Evolution of k_x and k_c. (b) Evolution of radius.

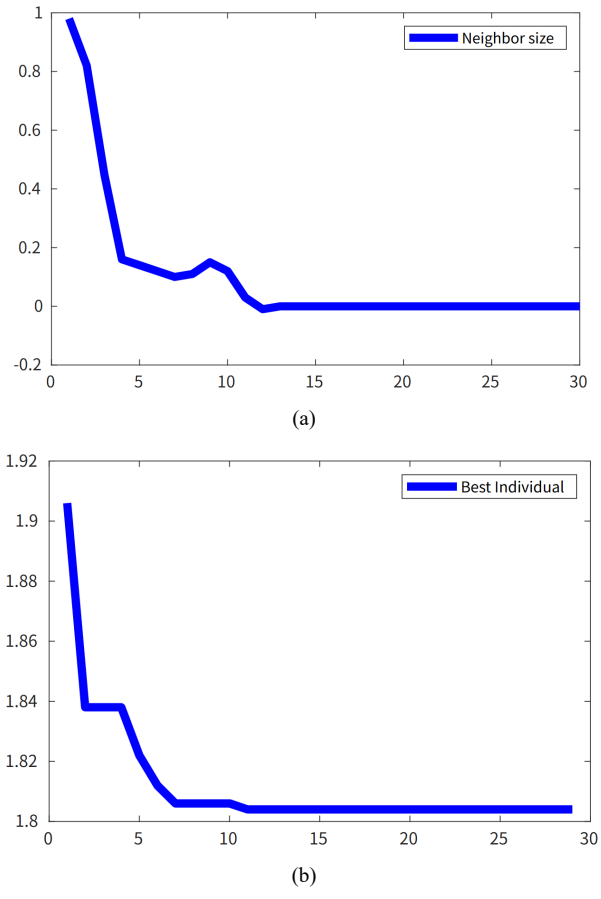


Fig. 10. Evolution of the parameters in 30 iterations. (a) *Reversible* parameters, (b) Evolution of best individuals in the learning curve.

B. Discussion

Looking at Fig. 10(b), it can be stated that the model requires less than fifteen iterations to reach convergence, which indicates that once the appropriate structure is determined, the PSO algorithm can estimate the best parameters of the model rapidly.

C. Model Complexity

For each input (color reference and depth image), it was processed pixel-by-pixel in the interpolation stage. If N is the number of image pixels, $B_M(x)$ is a neighborhood of M pixels around each pixel x, and the central pixel is compared with every pixel in B, we can say that the order of operations related to our method is $\mathcal{O}$ (N × M).

We have empirically evaluated the above statement by processing images from the KITTI dataset and measuring the processing time. In Table 1, we present the processing times over 10 iterations for images in the training set. The Table shows the processing time for different sizes (r) of the neighbor $N(x)$, and the average processing time ± standard deviation.

Table 1. Processing time for different images and neighborhood size

Image Number	Time processing per image miliseconds			
	$r = 1$	$r = 2$	$r = 3$	$r = 4$
Image1	119	340	640	1059
Image2	110	315	622	1029
Image3	89	254	500	827
Image4	110	293	579	960
Average	107.0±12.7	300.5±36.5	585.3±62.3	965.8±101.5

The number of points (N) in the neighborhood depends on the geometry of the neighborhood. In our case, we considered a square neighborhood, i.e., $N = (2 \times r + 1)^2$, which is a quadratic relation.

Fig. 11 illustrates the relationship between the average processing time and the neighborhood size r, revealing a quadratic trend as previously discussed. It is worth noting that, in all cases, the proportion of points filled remains approximately constant at around 95% of the image.

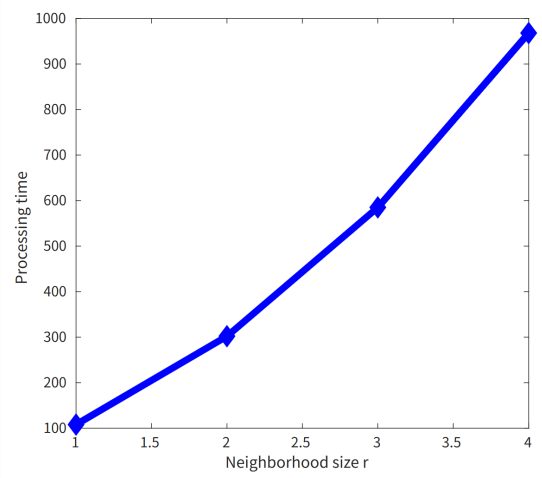


Fig. 11. Empirical average processing time in milliseconds for neighborhood size r.

D. Experiments

This section provides a brief description of our experiments addressed in this work.

Experiment 1: We trained the model using four images from the KITTI dataset and their respective sparse depth map and ground truth. The remaining 996 images were considered to evaluate the trained model.

Table 2 shows the result obtained by our model in the training set after 30 iterations. 40 individuals were used for the PSO.

Table 2. Obtained results in the training stage

Model	*MSE + MAE*
Our Proposal	1.8043 m

Experiment 2: We established a second realization of our model using the NYU-Depth V2 dataset available in Ref. [3]. Four images from this dataset, along with their corresponding ground truth, were utilized to train our model.

We subsampled the available depth data, then, we completed the missing data using our model and finally, the performance is computed. In this work, we subsampled the data by 4× (one data every other 4×4 square pixels).

Table 3 presents the results obtained by the model in this second dataset.

Table 3. Obtained results in the training stage using NYU-Depth V2 dataset

Model	*MSE + MAE*
Our Proposal 4×	8.6643 m

E. Discussion

In Experiments 1 and 2, we evaluated the performance of our model using the combined metric *MSE* + *MAE*, computed with the best hyperparameters obtained through the PSO algorithm. This choice of metric provides a balanced view of the model's prediction accuracy by capturing both the average Magnitude of Errors (MAE) and the Squared Deviations (MSE), which are particularly relevant in depth estimation tasks where both small-scale and large-scale errors are critical.

To evaluate the generalization capability of the model, we trained it using only four images from each dataset, despite the fact that the datasets contain between 20,000 and 80,000 images. This minimal training setup was chosen deliberately to demonstrate the robustness of the model in low-data regimes, which is a common challenge in real-world applications where labeled depth data is scarce or expensive to obtain. Interestingly, our results show that the model achieves competitive performance even under such limited supervision, suggesting strong inherent generalization properties.

In addition, preliminary experiments indicated that increasing the number of training images beyond this small subset led to indications of overfitting, and the model exhibited reduced performance on unseen samples. This behavior reinforces our design choice, highlighting the effectiveness of the PSO-tuned architecture and the model's capacity to learn meaningful representations from minimal data.

IV. Results and Discussion

This section provides a brief discussion of the numerical and qualitative results obtained by our proposed model.

A. Numerical Results

To evaluate the performance of the trained model using

996 images of the KITTI dataset, the MSE between the reconstructed depth map and the ground truth was computed. Table 4 shows the obtained results.

Table 4. Obtained results of the model in KITTI data set

Model	MSE
Ours	1.127 m
Model in Ref. [12]	1.643 m
PDC [15]	1.287 m

As shown in Table 4, our model demonstrates superior performance compared to several contemporary models. It offers a cost-effective implementation solution suitable for applications such as embedded systems. However, it is important to note that our model does not compete with more complex deep learning implementations. However, it is easy and quick to implement.

Table 5 shows the numerical results of our model in the upsampling task for the NYU-Depth V2 dataset. Our model outperforms some contemporary models such as shown in Ref. [16] and performs similarly to other models such as shown in Ref. [17].

Table 5. Obtained results of the model using NYU-Depth V2 dataset

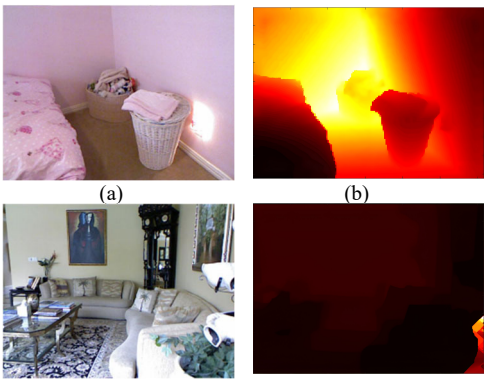
Model	MSE
Ours	5.80 cm
TGV [16]	6.98 cm
Ham [17]	5.27 cm

B. Qualitative Results

Fig. 12 shows two examples from the NYU-Depth V2 dataset, one in which our model fails to complete the depth data, and another in which our model performs the best.

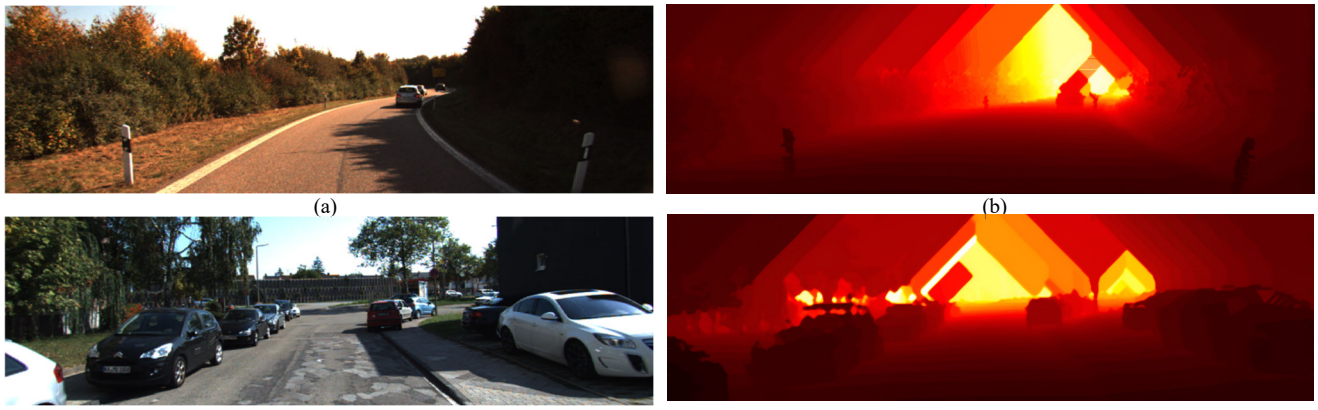
In Fig. 12(b), we can clearly observe the reconstructed basket and the towel on top of it. In Fig. 12(d) we can observe that to the right of the recovered depth map, the largest values of the depth data were diffused to the completed map.

Fig. 13 shows an example of the best and worst results obtained from our model using the KITTI dataset. Fig. 13(a) and (c) show the reference color images. Fig. 13(b) and (d) show their corresponding completed depth maps.



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

Fig. 12. Obtained results using NYU-Depth V2 dataset. (a) Original color reference image. (b) Best result obtained by our model. (c) Color reference image. (d) Worst result obtained by our model.



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

Fig. 13. Qualitative results for KITTI dataset. (a) Original color reference image. (b) Best result obtained by our model. (c) Color reference image. (d) Worst result obtained by our model.

C. Validation via Monte Carlo

We conducted experiments to evaluate the variability in the estimated performance of our proposal across different training sets. In our approach, we used four randomly selected reference images, their sparse depth maps, and their corresponding ground truth.

In these experiments, we used the KITTI dataset and constructed five subsets with the same characteristics as the set described above. We will refer to these subsets as F1, F2, F3, F4, F5, and the remaining data were assigned to the sets L1, L2, L3, L4, and L5, respectively.

Table 6. Training and validation sets for the validation experiment

Set	Experiment				
	1	2	3	4	5
Training	F1	F2	F3	F4	F5
Validation	L1	L2	L3	L4	L5

Table 6 shows the sets considered for training and validation for each experiment. The four images of the training set were randomly selected and the validation set comprises 996 reference images, along with their corresponding depth data, and ground truth.

Table 7 presents the performance metrics obtained during the training phase for each training set. The model achieved an average value of $MSE + MAE = 1.6558$ with a variance of 0.2685. Additionally, three results are close to $MSE + MAE \approx 1.8000$ m, which is in the neighborhood of the value reported in this manuscript. In contrast, two results are around 1.3000 m, these values are far from the ones reported in Table 2. These findings indicate the variability in performance across different training sets. However, the most critical performance metric is that of the validation set, as this value is used to evaluate the generalization capabilities of our

proposed model.

The results obtained from the validation set are presented in Table 8, which summarizes our findings from this validation process. The results indicate a small variability in the *MSE* value, with σ_{MSE} = 0.0156, which represents 1.4% of the average value. Additionally, for *MAE*, we recorded σ_{MAE}= 0.0099, representing 3.1% of the average value.

Table 7. Model performance evaluated in each training set

Training and Evaluation Subset	*MSE* + *MAE*
F1	1.2561
F2	1.8016
F3	1.8087
F4	1.7425
F5	1.3687
Average	1.6558±0.2685

Table 8. Model performance evaluated on its corresponding validation set

Training	Validation	*MSE*	*MAE*
F1	L1	1.1291	0.3115
F2	L2	1.1638	0.3334
F3	L3	1.1251	0.3135
F4	L4	1.1325	0.3086
F5	L5	1.1306	0.3167
Average		1.1362±0.0156	0.3167±0.0099

D. Discussion

This work presented a depth completion model based on a variable structure framework. The architecture of the model is dynamically determined during the training phase by optimizing key structural parameters, such as the number of convolutional filters, pipeline sequences, and the number of iterations. These parameters are estimated within the training process, enabling the model to adapt its structure for better performance. This flexible design enables the proposed model the capacity to better fit varying data characteristics and improves overall depth prediction accuracy.

The depth interpolation component of the model leverages the infinity Laplacian operator. This interpolator is particularly attractive due to its computational simplicity: it relies on a weighted average of the minimum and maximum values of a function defined in a neighborhood. Such simplicity makes it well-suited for deployment in resource-constrained environments, such as embedded systems with limited processing power and memory.

To evaluate the effectiveness of the proposed model, we conducted experiments using two datasets: one featuring outdoor urban scenes (KITTI), and another composed of indoor scenes (NYU-Depth V2). Both datasets include RGB images paired with corresponding depth maps acquired by active sensors.

We performed validation experiments using Monte Carlo, defining five training sets, and used the remaining data for validation. We observe that the results in the validation set exhibit small variability, less than 2% for *MSE* and less than 4% for *MAE*. The training dataset was randomly selected from the KITTI data set, covering different situations: bikers, pedestrians, cars, and trucks, and illumination conditions, including shadows, reflections, saturation, and others. Given the small variability and the random nature of the selected images, we infer that the performance of the model will remain almost the same for the depth completion task in similar conditions, i.e., color reference images, LiDAR sensor, in an urban context, with natural illumination, non-overcrowded traffic, vegetation on the edge of the road, and relatively slow velocity.

Regarding the processing time estimation, we estimated the processing time using our implementation which is not optimal and can be improved, increasing the efficiency of the model.

The complexity of the model is in fact quadratic with respect to the radius of a square neighborhood around a central point x. A common challenge across both datasets is the presence of reflective surfaces and transparent objects, where depth sensors typically fail to capture accurate data. In scenarios involving reflective surfaces, our model achieved satisfactory performance. On the other hand, in the presence of transparent objects, such as glass, the model's performance degraded considerably. The most pronounced errors were observed in the KITTI dataset, where transparent surfaces led to the worst depth completion results.

To address this limitation, future model iterations will incorporate transparent object detection, such as glass segmentation, and semantic scene understanding. These enhancements will allow the model to identify and mask problematic regions, enabling more accurate data propagation around object boundaries and improving robustness in real-world scenarios.

V. Conclusion

In this paper, we present a detailed variable-pipeline model whose structure is estimated during training. The model was applied to the depth completion problem. We empirically demonstrate that this variable structure improves the performance of a convolutional-interpolator model. The performed experiments, especially the validation using Monte Carlo, show that the performance evaluation exhibits small variability (less than 2%), helping us claim the robustness of its generalization capabilities. Future work will consider incorporating other interpolation methods, such as non-local means.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

V.L. conducted the research; V.L. A.D.C and F.C. wrote the paper; all authors had approved the final version.

References

[1] Z. Teed and J. Deng, "RAFT: Recurrent all-pairs field transform for optical flow," in *Proc. European Conference on Computer Vision*, 2020, pp. 402–419. https://doi.org/10.1007/978-3-030-58536-5_24

[2] J. Uhrig, N. Schneider, L. Schneider *et al.*, "Sparsity invariant CNNs," in *Proc. 2017 International Conference on 3D Vision (3DV)*, 2017, pp. 11–20. doi: 10.1109/3DV.2017.00012

[3] N. Silberman, D. Hoiem, P. Kohli *et al.*, "Indoor segmentation and support inference from RGBD images," in *Proc. the European Conference on Computer Vision (ECCV)*, 2012, pp. 746–760.

[4] D. A. Montecino, C. A. Perez, and K. W. Bowyer, "Two-level genetic algorithm for evolving convolutional neural networks for pattern recognition," *IEEE Access*, vol. 9, 126856–126872, 2021. doi: 10.1109/ACCESS.2021.3111175

[5] J. Z. Esquivel, A. C. Vargas, P. Lopez *et al.*, "Adaptive convolutional kernels," in *Proc. IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, 2019, pp. 1998–2005.

[6] C. Cortes, X. Gonzalvo, V. Kuznetsov *et al.*, "AdaNet: Adaptive structural learning of artificial neural networks," in *Proc. the International Conference on Machine Learning*, 2017, pp. 874–883.

[7] Y. Li, Z. Zhang, X. Li *et al.*, "Depth image restoration based on uncertainty-guided priority optimization," *Journal of Electronic Imaging*, vol. 34, no. 1, 2025. doi: 10.1117/1.jei.34.1.013022

[8] V. Lazcano and F. Calderero, "Depth completion with anisotropic metric, convolutional stages, and infinity Laplacian," *Applied Sciences*, vol. 14, no. 11, 4514, 2024. doi: 10.3390/app14114514

[9] Z. Yue, J. Wang, and C. C. Loy, "Efficient diffusion model for image restoration by residual shifting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 47, no. 1, pp. 116–130, 2025. doi: 10.1109/TPAMI.2024.3461721

[10] M. Xiong, Z. Zhang, J. Liu *et al.*, "Self-supervised depth completion with multi-view geometric constraints," *IET Image Processing*, vol. 17, no. 11, pp. 3095–3105, 2023. doi: 10.1049/ipr2.12834

[11] W. Zhao, C. Jung, and J. Kim, "Deep sparse depth completion using multi-affinity matrix," *IEEE Access*, vol. 11, 2023. doi: 10.1109/access.2023.3295133

[12] V. Lazcano, F. Calderero, and C. Ballester, "Comparing different metrics on an anisotropic depth completion model," *International Journal of Hybrid Intelligent Systems*, vol. 17, no. 3–4, pp. 87–99, 2021. doi: 10.3233/HIS-210006

[13] V. Caselles, L. Igual, and O. Sander, "An axiomatic approach to scalar data interpolation on surfaces," *Numerische Mathematik*, vol. 102, no. 3, pp. 383–411, 2006.

[14] J. J. Manfredi, A. M. Oberman, and A. P. Svirodov, "Nonlinear elliptic partial differential equations and p-harmonic functions on graphs," *Differential and Integral Equations*, vol. 28, no. 1–2, pp. 79–102, 2015.

[15] D. Teutscher, P. Mangat, and O. Wassermueller, "PDC: Piecewise depth completion utilizing superpixels," in *Proc. 2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 2752–2758.

[16] D. Ferstl, C. Reinbacher, R. Ranftl *et al.*, "Image guided depth upsampling using anisotropic total generalized variation," in *Proc. IEEE International Conference on Computer Vision*, 2013, pp. 993–1000.

[17] B. Ham, M. Cho, and J. Ponce, "Robust image filtering using joint static and dynamic guidance," in *Proc. Computer Vision and Pattern Recognition*, 2015, pp. 4823–4831.