On the Relevance of Patch-based Extraction Methods for Monocular Depth Estimation

Pasquale Coscia, Antonio Fusillo*, Angelo Genovese, Vincenzo Piuri, Fabio Scotti

Department of Computer Science, Università degli Studi di Milano, 20133, Milan, Italy

Abstract

Scene geometry estimation from images plays a key role in robotics, augmented reality, and autonomous systems. In particular, Monocular Depth Estimation (MDE) focuses on predicting depth using a single RGB image, avoiding the need for expensive sensors. State-of-the-art approaches use deep learning models for MDE while processing images as a whole, sub-optimally exploiting their spatial information. A recent research direction focuses on smaller image patches, as depth information varies across different regions of an image. This approach reduces model complexity and improves performance by capturing finer spatial details. From this perspective, we propose a novel warp patch-based extraction method which corrects perspective camera distortions, and employ it in tailored training and inference pipelines. Our experimental results show that our patch-based approach outperforms its full-image-trained counterpart and the classical crop patch-based extraction. With our technique, we obtain a general performance enhancements over recent state-of-the-art models. Code will be available at https://github.com/AntonioFusillo/PatchMDE.

Keywords: Monocular Depth Estimation (MDE), Autonomous Driving (AD), Patch-based Approach, Single Image Depth Estimation (SIDE), Metric Depth.

1. Introduction

Understanding the geometric structure of the environment from visual inputs is a long-standing goal in computer vision. Accurate depth perception is crucial in numerous real-world applications such as robotics, augmented reality, and autonomous driving, where machines must navigate and interact with complex, dynamic scenes. Among the various techniques for depth sensing,

*Corresponding author

Email addresses: pasquale.coscia@unimi.it (Pasquale Coscia), antonio.fusillo@unimi.it (Antonio Fusillo), angelo.genovese.unimi.it (Angelo Genovese), vincenzo.piuri@unimi.it (Vincenzo Piuri), fabio.scotti@unimi.it (Fabio Scotti)

Monocular Depth Estimation (MDE) [1], also known as Single Image Depth Estimation (SIDE), has gained particular attention due to its low cost, wide applicability, and compatibility with compact camera setups. In fact, despite their effectiveness, physical depth sensors can be unreliable in certain situations, such as when reflective surfaces are present. Stereo rigs also pose challenges, as they require frequent recalibration. On the other hand, MDE is a geometric reconstruction task that aims to estimate the depth of a scene from a single color image, leveraging on scene context to resolve scale ambiguities. Recent advancements in deep learning have enabled neural networks to understand the visual information with remarkable accuracy and efficiency [2], making MDE a suitable alternative for scene geometry reconstruction.

State-of-the-art approaches for MDE [3, 4, 5, 1] process the entire input image, rather than focusing on smaller patches to be treated independently. While full-image approaches are feasible with today's computational power, they have several drawbacks. For example, predictions may depend on complex relationships across the entire image, which limits parallelization, a key advantage for real-time applications. Additionally, applying a full-image model to image sizes different from those used during training may lead to inaccurate metric depth predictions and require further fine-tuning. Instead, image patches allow for greater flexibility, as they are independent of the aspect ratio and can be processed separately, enhancing both efficiency and applicability. Various patch-based approaches are proposed in the MDE literature. Some authors focus on designing new architectures that better exploit local information [6, 7, 8]; others tackle on the problem of high-resolution affine depth estimation [9]; and there are works on 360-images depth estimation using tangent projections [10, 11]. However, despite the use of patch-based approaches for MDE, no thorough investigation has been conducted to understand which is the best way to extract patches and the effect of processing them independently from each other.

In this paper, motivated by the perspective distortions that cropping patches introduces, we propose a novel camera-consistent patch extraction procedure and inference pipeline to compose the global prediction of an image using patch predictions. Our method consists of the following steps: *i)* sampling patches via a grid; *ii)* extracting patches with a perspective transformation (warp) to correct perspective distortions; *iii)* generating depth predictions for each patch; *iv)* re-projecting depth maps onto the image plane; and *v)* merging estimates with a weighted average in overlapping regions. We train various neural architecture using either full images, cropped patches or warped patches and compare their performance. We find that patch-based methods perform better than the ones based on the full image and that warping patches in a camera-consistent way is beneficial and improves accuracy over simple cropping, even when applied to state-of-the-art MDE architectures.

The contribution of our approach is three-fold:

1. we propose an effective methodology for extracting patches while maintaining consistent camera intrinsic parameters for MDE.
2. We design a simple, yet effective, inference pipeline to estimate depth

using patch-based predictions.

3. We show that our pipeline can boost state-of-the-art models.

The paper is organized as follows. Section 2 discusses related previous works. Section 3 introduces the pinhole camera model and the used notation. Section 4 presents a detailed explanation of the proposed inference and training pipelines. Section 5 describes the experiments conducted and discusses the results. Finally, Section 6 concludes the work and suggests future directions.

2. Related Work

MDE has been addressed using hand-crafted features or, more recently, deep learning models. Below, we review the key methods in these areas, along with the main approaches that propose patch-based solutions.

Hand-crafted Features. Early works address MDE by designing hand-crafted features and pipelines that rely on strong assumptions [12]. For example, Hoeim et al. develop an algorithm [13] to create a cardboard-like model of the image, decomposing outdoor scenes into three elements: the ground, vertical objects, and the sky. DepthTransfer [14] introduces a non-parametric pipeline that constructs the final prediction by warping ground-truth depth maps from nearest-neighbor images in the dataset through scene-flow estimates.

Supervised Monocular Depth Estimation. MDE can be approached with different kinds of supervision signals, depending on the type of depth information being estimated. It can be solved as a *metric* (or regression) problem assigning a depth value to each image pixel corresponding to the depth of the object in the camera coordinate system. In this case, the aim is to estimate depth measurements w.r.t. some physical units. In this category, Eigen et al. [15] propose the first neural network-based approach, framing metric MDE as a regression problem. They train a simple convolutional neural network with a scale-invariant loss function to address scale ambiguities in depth regression. Laina et al. [16] improve upon this approach by using a ResNet-based model [17]. Some works also formulate MDE as a classification problem by binning the depth range [18, 19, 20, 21, 3].

Alternatively, MDE can be framed as a *relative* problem [22, 23, 24], as gathering ground-truth metric data is expensive, where a depth map is used only for relative depth comparisons within an image (i.e., determining which points are farther apart). Suitable datasets for this approach [22], collected via crowd-sourcing, provide relative depth annotations for pairs of random points.

Finally, MDE can also be approached as an *affine* problem, as proposed by Ranftl et al. in [25]. Their approach unifies many available metric datasets that present different camera settings by working in disparity space up to an affine transformation. In this way, they facilitate the scaling of supervised model training. Subsequent works build upon these ideas, enabling the training of larger models [26, 27, 5].

Patch-based Monocular Depth Estimation. Miangoleh et al. [9] address the challenge of affine MDE for high-resolution images proposing a pipeline for merging low- and high-resolution predictions. This pipeline processes scaled versions of the input image and its patches to generate predictions. While they work in an affine setting, we tackle metric depth regression and focus on keeping consistent camera parameters. To avoid relying on pre-trained backbones, Li et al. [6] design an end-to-end architecture considering a global scale-aware estimation network, a patch-wise depth prediction network, and a guided-fusion network. Li et al. [7] additionally include residual corrections for both coarse- and fine-grained estimations. Both [6] and [7] treat patches as additional input sources. Instead, we study how to solve MDE when predictions are exclusively patch-wise.

Several patch-based approaches can also work with depth estimation in 360° imaging. Rey-Area et al. [11] apply a pre-trained MDE to tangent projections of an omnidirectional image and merge the predictions through a global optimization algorithm. Similarly to our approach, tangent patches are all captured by the same camera. Li et al. [28] train an architecture that predicts patch-wise depth estimates using an encoding of the patch positions on the camera sphere and use a transformer that aggregates global information. Finally, an iterative refinement step computes the final depth map with a weighted average based on the estimated confidence maps for each patch. Chen et al. [10] process the equirectangular projection of the image using a U-net-like architecture, injecting the local depth estimates in the decoding phase. An additional head to the decoder predicts sky segmentation maps. Zhang et al. [29] propose a hierarchical depth normalization procedure based on image sub-regions for better learning of affine depth while Yan et al. [8] design a patch-wise attention module as a refinement layer after following an encoder-decoder architecture.

The works that most align with our are those of [9] and [11], but while they consider an affine MDE setting with pre-trained models, we predict metric depth with an architecture trained from scratch exclusively on patches.

3. Preliminaries

This section provides a brief review of the pinhole camera model and introduces the notation for the ensuing methodology.

Pinhole Camera. A pinhole perspective camera is described by a tuple $\mathcal{C} = (\mathbf{K}, [\mathbf{R} \,|\, \mathbf{t}])$, where $\mathbf{K}$ is the intrinsic parameter matrix, while $\mathbf{R}$ and $\mathbf{t}$ are the rotation and translation matrices, representing the extrinsic parameters that describe the camera's position and orientation relative to the world coordinate system. The intrinsic matrix describes an invertible transformation that encodes the camera's focal length (f) and principal point (p_x, p_y), defining the scene projection geometry of the scene. The extrinsic parameters describe a rigid transformation mapping world coordinates to the camera reference frame.

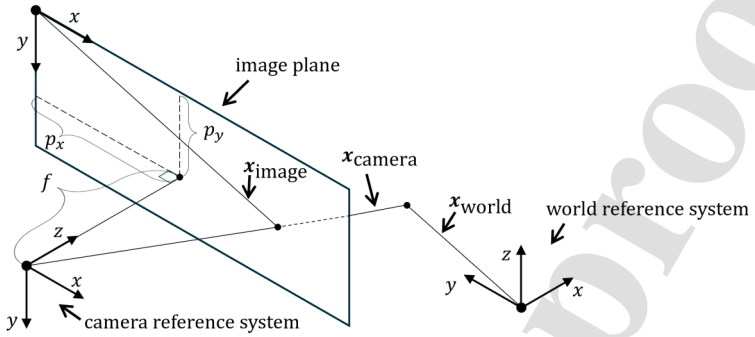


Figure 1: Pinhole camera model.

Matrix $\mathbf{K}$ is typically expressed as follows:

$$\mathbf{K} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

In the pinhole camera model, a point in the world reference system $\mathbf{x}_{\text{world}}$ is first mapped to the camera reference frame as $\mathbf{x}_{\text{camera}} = [\mathbf{R} \,|\, \mathbf{t}]\ \mathbf{x}_{\text{world}}$ and then projected onto the image plane via $\mathbf{x}_{\text{image}} = \mathbf{K}\, \mathbf{x}_{\text{camera}}$.

For the sake of simplicity, we omit explicit vector homogeneization and de-homogeneization, assuming a natural representation of each object and applying the necessary transformations implicitly. Specifically, $\mathbf{x}_{\text{world}}$ is represented as a 3D vector, which is augmented with a homogeneous coordinate (appending a 1) to be compatible with $[\mathbf{R} \,|\, \mathbf{t}] \in \mathbb{R}^{3\times 4}$. Similarly, $\mathbf{x}_{\text{image}}$ is a 2D vector, obtained by de-homogenizing the projected results, that is dividing the first two coordinates by the third (non-zero) component.

Given $\mathbf{x}_{\text{image}}$ and the depth z of its corresponding $\mathbf{x}_{\text{camera}}$, the back-projection to the camera reference system is defined as $\mathbf{x}_{\text{camera}} = z\, \mathbf{K}^{-1}\, \mathbf{x}_{\text{image}}$. It is also possible to obtain $\mathbf{x}_{\text{world}}$ using $\mathbf{x}_{\text{world}} = [\mathbf{R} \,|\, \mathbf{t}]^{-1}\ \mathbf{x}_{\text{camera}}$, where $[\mathbf{R} \,|\, \mathbf{t}]^{-1} = \left[\mathbf{R}^{-1} \,|\, -\mathbf{R}^{-1}\mathbf{t}\right]$.

Image Re-projection. Given a second camera $\mathcal{C}' = (\mathbf{K}', [\mathbf{R}' \,|\, \mathbf{t}'])$, a point $\mathbf{x}_{\text{image}}$ in the image plane of $\mathcal{C}$ is mapped to its corresponding location $\mathbf{x}'_{\text{image}}$ in the image plane of $\mathcal{C}'$ as follows:

$$\mathbf{x}'_{\text{image}} = \mathbf{K}'\ \ [\mathbf{R}' \,|\mathbf{t}']\ \ [\mathbf{R} \,|\, \mathbf{t}]^{-1}\ \ z_{\mathcal{C}}\, \mathbf{K}^{-1}\ \mathbf{x}_{\text{image}} \tag{2}$$

Here, $z_{\mathcal{C}}$ denotes the depth of $\mathbf{x}_{\text{image}}$ in the reference system of $\mathcal{C}$. If the second camera is purely rotated relative to the first, such that $\mathbf{x}'_{\text{camera}} = \tilde{\mathbf{R}}\, \mathbf{x}_{\text{camera}}$ for some rotation matrix $\tilde{\mathbf{R}}$, then $z_{\mathcal{C}}$ can be an arbitrary non-zero value since

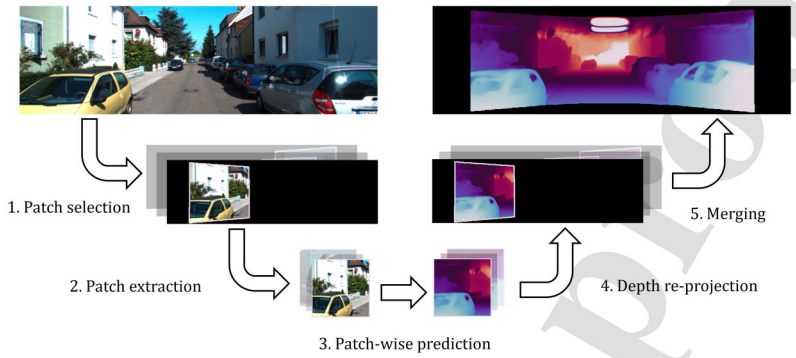


Figure 2: Outline of the proposed patch-based inference pipeline for depth estimation. The pipeline processes an input image by *i)* selecting patch centers, *ii)* extracting patches, *iii)* estimating depth for each patch, *iv)* re-projecting the depth maps, and finally *v)* merging the depth predictions into a unified output.

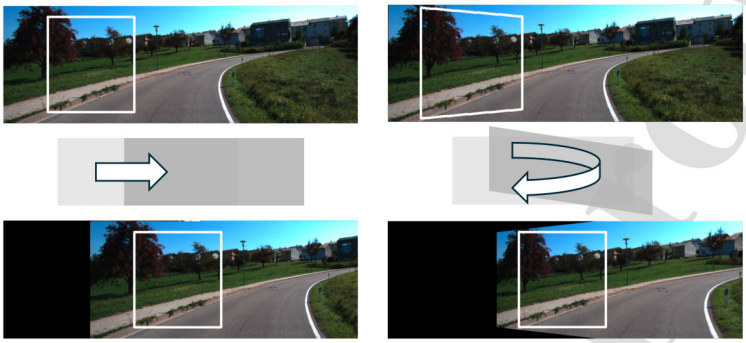
projection centers of $\mathcal{C}$ and $\mathcal{C}'$ are identical. This assumption holds throughout this paper. Based on this formulation, an image acquired by $\mathcal{C}$ is denoted as $I_{\mathcal{C}}$, while its re-projection w.r.t. $\mathcal{C}'$ is written as $I_{\mathcal{C}\to\mathcal{C}'}$. This re-projected image can be computed using standard image transformation techniques.

With the above mathematical formulation, we aim to extract patches from each input image using a specific transformation method (e.g., translation or warping), perform our predictions, and re-project the results onto the original image plane.

4. Methodology

To study the effects of different patch-extraction methods, we train a neural network on a patch-based dataset derived from a wide-aspect-ratio dataset. The patch-based dataset is obtained either cropping or warping patches out of the images of the original dataset. The trained patch-model is then employed for inference on full images through our pipeline. This section describes in detail our method. In particular, the proposed inference pipeline is depicted in Fig. 2 and consists of the following steps:

1. *Patch selection:* Given an input image I, we select a finite set of patch centers $\{(x_c^n, y_c^n)\}_{n=1}^N$.
2. *Patch extraction:* For each center, we extract a patch P_n centered at (x_c^n, y_c^n) by warping I.
3. *Patch-wise prediction:* For each patch P_n, we estimate the corresponding depth map Z_n using a neural network ϕ_θ.
4. *Depth re-projection:* Each depth map Z_n is re-projected onto the image plane of I.



(a) (b)

Figure 3: (a) Patch extraction via image translation (*crop-based*). (b) Patch extraction via perspective transformation (*warp-based*).

5. *Merging:* The overlapping depth patches Z_n are merged to obtain the final depth prediction Z.

4.1. Patch Selection

To ensure a global coverage of the input image I, we sample a finite set of patch centers $\mathcal{S}(I) = \{(x_c^n, y_c^n)\}_{n=1}^N$ from the continuous domain $[0, W) \times [0, H)$, where H, W are the height and width of I, respectively. This is necessary because each patch represents only a portion of the input image. Each point defines the center of the n-th patch P_n to be extracted.

The selection procedure samples patch centers along the horizontal direction of the image. We cover the image along the vertical direction considering tall patches, since the vertical context is known to be important for a successful depth estimation [30]. During training, patch centers are randomly sampled. During inference, a fixed number of uniformly spaced patch centers is considered.

4.2. Patch Extraction

To extract a patch, we consider two distinct techniques, illustrated in Fig. 3. Conventional crop-based extraction is formally reviewed in 4.2.1, while the more precise warp-based extraction is introduced in 4.2.2. In the following, we primarily discuss their differences in terms of camera-consistency, depicted in Fig. 3. Examples of crop- and warp-based extracted patches are shown in Fig. A.13.

4.2.1. Crop-based Patch Extraction

Let $I_\mathcal{C}$ be an image acquired with a known camera $\mathcal{C} = (\mathbf{K}, [\mathbf{R}, \mathbf{t}])$. If H and W denote the height and width of $I_\mathcal{C}$, then $I_\mathcal{C}(x, y)$ is defined only within the valid pixel range: $0 \leq x < W$, $0 \leq y < H$. Patches of a perspective image can

Figure 4: Examples of patches sampled from the extremes of the image using either crop- (top row) or warp-based (bottom row) extraction method. It is possible to observe that patches extracted with the warp-based method exhibit significantly less perspective distortions.

be extracted by a cropping operation characterized by a patch center (x_c, y_c) and a shape (h, w) so that the resulting patch P is defined as:

$$P(u,v) = I_{\mathcal{C}}\left(u - \frac{w}{2} + x_c,\, v - \frac{h}{2} + y_c\right) \quad \text{for} \;\; 0 \le u < w\,,\; 0 \le v < h \tag{3}$$

If any of the following conditions hold:

$$x_c \ge W - \frac{w}{2}, \quad x_c < \frac{w}{2}, \quad y_c \ge H - \frac{h}{2}, \quad \text{or} \quad y_c < \frac{h}{2} \tag{4}$$

then $P(u,v)$ is not defined, requiring a padding operation on $I_{\mathcal{C}}$ to extend its definition.

To model the distortion introduced by the crop-based patch extraction, we consider the relationship between patch coordinates $\mathbf{x}_{\text{patch}}$ and image coordinates $\mathbf{x}_{\text{image}}$ as:

$$\mathbf{x}_{\text{patch}} = \underbrace{\begin{bmatrix} 1 & 0 & \frac{w}{2} - x_c \\ 0 & 1 & \frac{h}{2} - y_c \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{T}} \mathbf{x}_{\text{image}} \tag{5}$$

which leads to the following:

$$\mathbf{x}_{\text{patch}} \;=\; \mathbf{T}\,\mathbf{x}_{\text{image}} \;=\; \mathbf{T}\,(\mathbf{K}\,\mathbf{x}_{\text{camera}}) = (\underbrace{\mathbf{T}\,\mathbf{K}}_{\mathbf{K}'})\,\mathbf{x}_{\text{camera}} \tag{6}$$

Therefore, patch P can be considered as acquired by a camera $\mathcal{C}' = (\mathbf{K}', [\mathbf{R},\, \mathbf{t}])$, i.e., $P = I_{\mathcal{C}\to\mathcal{C}'}$. Notice that, computationally, this corresponds to an image

translation, as illustrated in Fig. 3(a). The internal parameters of the new defined camera are then expressed as follows:

$$\mathbf{K}' = \begin{bmatrix} f & 0 & p_x + \frac{w}{2} - x_c \\ 0 & f & p_y + \frac{h}{2} - y_c \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

According to Eq. 7, the acquisition geometry is dependent on the patch center (x_c, y_c), introducing distortions in objects appearances based on their location in the image. This is particularly relevant for images with a large field of view.

4.2.2. Warp-based Patch Extraction

To ensure consistent appearances of objects across scenes, all patch cameras should share internal camera parameters. We fix $\mathbf{K}'$ to be:

$$\mathbf{K}' = \begin{bmatrix} f' & 0 & \frac{w}{2} \\ 0 & f' & \frac{h}{2} \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

where h, w are the fixed height and width of the patches and f' is a fixed focal length value.

We achieve consistency in the acquisition geometry by making the camera point to the center of each patch by means of a rotation. Note that translating the camera is not feasible, as it would require new visual information not present in the original image. In particular, we consider an image $I_{\mathcal{C}}$ and its acquisition camera $\mathcal{C} = (\mathbf{K}, [\mathbf{R}, \mathbf{t}])$, with (x_c, y_c) as the patch center in the $\mathcal{C}$ image plane. To compute the rotations, we analyze the angles formed by the back-projected ray of the patch center, i.e., the line passing through the camera center and the patch center on the image plane, relative to the camera axes:

$$\theta_x = \tan^{-1}\left(\frac{x_c - p_x}{\sqrt{f^2 + (y_c - p_y)^2}}\right), \quad \theta_y = \tan^{-1}\left(\frac{y_c - p_y}{f}\right) \tag{9}$$

Using θ_x and θ_y, we obtain the rotation matrix to orient the camera towards the chosen patch center:

$$\tilde{\mathbf{R}} = \begin{bmatrix} cos(\theta_x) & 0 & -sin(\theta_x) \\ 0 & 1 & 0 \\ sin(\theta_x) & 0 & cos(\theta_x) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta_y) & -sin(\theta_y) \\ 0 & sin(\theta_y) & cos(\theta_y) \end{bmatrix} \tag{10}$$

Finally, the patch is defined as $P = I_{\mathcal{C}\to\mathcal{C}'}$ and its corresponding camera is $\mathcal{C}' = \left(\mathbf{K}', \left[\tilde{\mathbf{R}}\,\mathbf{R}, \tilde{\mathbf{R}}\,\mathbf{t}\right]\right)$, where $\mathbf{K}'$ is fixed independently of the location of the patch center, and only the extrinsic parameters depend on the position of the patch center. The resulting mapping from the $\mathcal{C}$ to the $\mathcal{C}'$ image planes is a homography. From a computational point of view, this corresponds to applying a perspective transformation to the image, as shown in Fig. 3(b). By means

of it, the ray of the patch center is mapped to the principal ray of the patch camera. In formulas:

$$\mathbf{K}' \tilde{\mathbf{R}} \mathbf{K}^{-1} \begin{bmatrix} x_c \\ y_c \end{bmatrix} = \frac{1}{2} \begin{bmatrix} w \\ h \end{bmatrix} \tag{11}$$

4.3. Patch-wise Prediction

Let $\mathcal{I}$ be an image dataset. A depth estimation model ϕ_θ is trained on patches extracted from various images $I \in \mathcal{I}$. Note that the estimation model can be any model capable of processing input images and returning depth estimations (e.g., a neural network trained in a supervised fashion), making our procedure architecture-agnostic.

Formally, from each image I captured by the camera $\mathcal{C}$, a set of patches centered at the points $\mathcal{S}(I)$ (as specified in Section 4.1) is extracted using an extraction method (either cropping or warping): $\mathcal{P}(I) = \{P_1, \dots, P_N\}$. From the ground-truth depth map Z^* of each image I, depth patches are extracted using the same centers and undergoing the same transformations. In this way, we obtain a set of image-depth patches $\mathcal{D}(I) = \{(P_n, Z_n^*)\}_{n=1}^N$.

More precisely, each patch is associated to a virtual camera $\mathcal{C}_n$ and $Z_n^* = Z_{\mathcal{C} \to \mathcal{C}_n}^*$. The training dataset is then defined as:

$$\mathcal{D}(\mathcal{I}) = \bigcup_{I \in \mathcal{I}} \mathcal{D}(I) \tag{12}$$

At inference time, to get a depth estimate of an image I, ϕ_θ is applied to each patch of $\mathcal{P}(I)$ independently to get predictions $Z_1, \dots, Z_N$ which are then re-projected onto the image and merged to form the final prediction Z.

It is worth noting that since the model is applied patch-wise, this step is fully parallelizable. Moreover, $\mathcal{P}(I)$ can be extracted for images of various shapes, as long as I is larger than the patches.

4.4. Depth Re-projection

As discussed in the previous subsection, given an image I captured by a camera $\mathcal{C}$, various patches P_n are extracted during inference, corresponding to (virtual) cameras $\mathcal{C}_n$. We then apply the trained patch MDE model ϕ_θ to each of these patches to obtain the predictions Z_n.

To re-project the predicted depth maps Z_n onto the image plane of I, we employ the following steps for each of them: *i)* we define a one-channel proxy image $\tilde{P}_{\mathcal{C}_n}$; *ii)* we warp the proxy image $\tilde{P}_{\mathcal{C}_n}$ to the camera $\mathcal{C}$ using Eq. 2. Notice that the last step cannot be applied directly to the depth map Z_n, as the depth values depend on the camera reference system.

We define $\tilde{P}_{\mathcal{C}_n}(u, v)$ using the following procedure: *i)* we back-project (u, v) to the $\mathcal{C}_n$ camera reference system at depth $Z_n(u, v)$, obtaining $\mathbf{x}_{\text{camera}}^{\mathcal{C}_n}$; *ii)* we express $\mathbf{x}_{\text{camera}}^{\mathcal{C}_n}$ w.r.t. $\mathcal{C}$, obtaining $\mathbf{x}_{\text{camera}}^{\mathcal{C}}$; *iii)* we assign the depth component of $\mathbf{x}_{\text{camera}}^{\mathcal{C}}$ to $\tilde{P}_{\mathcal{C}_n}(u, v)$.

Finally, $\tilde{P}_{\mathcal{C}_n \to \mathcal{C}}$ is the re-projection of Z_n to the $\mathcal{C}$ image plane (in the actual computation, nearest-neighbor interpolation is applied, and no anti-aliasing is used to preserve depth information).

The same algorithm, with patch and image roles inverted, is also applied for mapping ground-truth depth maps to patch camera spaces, to obtain $Z_n^* = Z^*_{\mathcal{C}\to\mathcal{C}_n}$ described in Subsection 4.3. An alternative approach consists in working directly with point clouds, but it is slower and prone to create holes in the final re-projected depth map.

4.5. Merging Procedure

After the model is trained, it can be used to make predictions on the entire image I. To this end, patches $P_1, \ldots, P_N$ are extracted from I, and predictions $Z_1, \ldots, Z_N$ are obtained by forwarding the model ϕ_θ on these patches:

$$Z_n(u_n, v_n) = \phi_\theta(P_n)(u_n, v_n), \quad n = 1, \ldots, N \tag{13}$$

Each point (u_n, v_n) in P_n corresponds to a point (x, y) in the image I, and we denote this relation as $(u_n, v_n) \equiv (x, y)$. Hence, each patch prediction assigns depth values to a subset of pixels in I. For each pixel, multiple values may be assigned from overlapping patches, with exactly one value coming from a single patch, or no value at all if the pixel does not belong to any of the considered patches. Pixels with no assigned value will remain empty, indicating that no depth prediction is available at such locations.

For the remaining pixels, the predicted value is computed as a weighted average of the various depth values. Specifically, for each position (u, v), a weight $c(u, v)$ is calculated using a Gaussian function that gradually diminishes towards the edges of the patch:

$$c(u, v) = exp\left(-\frac{(u - \frac{w}{2})^2 + (v - \frac{h}{2})^2}{w + h}\right) \tag{14}$$

The predicted depth value assigned to x, y in I is then:

$$Z(x, y) = \frac{\sum^n_{(u_n, v_n) \equiv (x,y)} c(u_n, v_n)\, Z_n(u_n, v_n)}{\sum^n_{(u_n, v_n) \equiv (x,y)} c(u_n, v_n)} \tag{15}$$

This procedure ensures smooth results with minimal complexity, avoiding artifacts from non-weighted averages due to scale differences between adjacent patches.

5. Experiments

This section outlines the experiments designed to evaluate our methodology. More specifically, subsection 5.1 details the experimental setup, including the dataset, architecture, training procedure, and evaluation methodology. Then, subsection 5.2, 5.3 and 5.4 present and discuss the baseline results and experiments obtained under various settings. Then, subsection 5.5 systematically analyzes the impact of the relevant hyperparameters. Finally, subsection 5.7 studies the computational cost of the overall pipeline.

5.1. Experimental Setup

We train a depth estimation model on patches extracted from a wide-aspect-ratio dataset, then evaluate our pipeline on a held-out test set and compare its performance to that of the same model, but trained on full images. We train a model for each Crop and Warp patch extraction techniques.

Dataset. We perform our experiments on the KITTI dataset [31], as all camera parameters are available, it is a well-established benchmark for metric depth estimation, and the camera field of view is sufficiently large for images to exhibit perspective distortions. This dataset contains various outdoor scenes captured by a stereo rig mounted on a car in daylight and in regular weather conditions with images of height and width of about 370×1200 pixels. It consists of $40,000$ samples, and we use the split described in [15] (Eigen split) to divide it into training, validation, and test subsets. Depth information is obtained using a LIDAR sensor mounted on the car, which provides sparse depth maps. Since the car is in motion and the projection centers of the depth sensor and the RGB camera differ, the raw data are noisy and exhibit various streaking artifacts [32]. We observe that a model trained on patches extracted from the raw data learns noisy information. For this reason we use the enhanced ground-truth from [32], publicly available on the KITTI official website[1], as it offers more accurate and denser depth maps.

Loss. We consider a variant [33] of the scale-invariant loss function [15] for training our model, defined in log-depth space. Specifically, let Z be the predicted depth map, whether from a patch or a full image, and Z^* be the corresponding ground-truth. Then:

$$\mathcal{L} = \alpha \sqrt{\frac{1}{K}\sum_{(x,y)} \left(log \frac{Z(x,y)}{Z^*(x,y)} \right)^2 - \lambda \left(\frac{1}{K}\sum_{(x,y)} log \frac{Z(x,y)}{Z^*(x,y)} \right)^2} \tag{16}$$

where (x, y) are pixels for which Z^* exists, K is the number of such pixels and α and λ are hyper-parameters set to 10 and 0.85.

Baseline Neural Architecture. We consider the neural network proposed in [23], which uses a ResNet-50 [17] backbone pre-trained on ImageNet and has 60M learnable parameters. Although this architecture is commonly used also for relative [23, 24] and affine [25] MDE, our results demonstrate its suitability for metric depth estimation. The model is trained to directly output log-depth values, with the final layer performing a convolution operation.

Following [25], the model is trained using the Adam optimizer [34] with a learning rate of 10^{-5} for the pre-trained encoder and 10^{-4} for the randomly initialized decoder. We decrease these learning rates by a multiplicative factor

[1] https://www.cvlibs.net/datasets/kitti/

of 0.9 every 8 epochs, for a total of 40 epochs. During inference, the output values are clamped to the range $[log(z_{min}), log(z_{max})]$ and then exponentiated to obtain positive depth values. As proposed in [15], for supervised metric depth estimation performed on the KITTI dataset, we set z_{min} and z_{max} to 0.001 and 80, respectively.

Baseline Experiment. We train the above architecture in four settings. *(Base-Half)*: the model is trained on full images at half the resolution (following [15]); *(Base-Full)*: the same as *Base-Half* but without halving the input image resolution; *(Base-Crop)*: the is model trained on patches cropped (4.2.1) from the input image; and *(Base-Warp)*: the model is trained on patches extracted via camera-consistent warping (4.2.2). Base-Crop and Base-Warp are trained on squared patches of size 300×300 pixels. The focal length f' of the patch cameras in Eq. 8 is set to 720 for Patch-MDE (Warp). We avoid extracting patches that require padding the initial image. During inference, we consider 9 patch-centers.

KITTI images have slightly different shapes across the dataset, but baseline models require fixed-size images as input. We resize them to 172×576 pixels for Base-Half and to 344×1152 pixels for Base-Full. During training, baseline models receive full-images, augmented as in [15], using random horizontal flip, random rotation, scaling, color jittering, and translation. By contrast, the only augmentation applied to patches is random horizontal flip and rotation. During inference, predictions are rescaled to the original resolution by means of nearest-neighbor interpolation and no anti-aliasing.

Evaluation. Evaluation is performed on the test set using improved KITTI ground-truth [32]. We use the standard metrics from [15], as follows: delta accuracies with thresholds $1.25^1, 1.25^2, 1.25^3$ (δ_1, δ_2, δ_3); absolute relative error (AbsRel); squared relative error (SqRel); root mean squared error (RMSE); root mean squared logarithmic error (RMSLE). We provide the definition of delta accuracies:

$$\delta_t = \frac{1}{N}\left|\left\{(x,y) \,:\, max\left(\frac{Z(x,y)}{Z^*(x,y)}, \frac{Z^*(x,y)}{Z(x,y)}\right) < 1.25^t\right\}\right|, \quad t = 1,2,3 \qquad (17)$$

where N is the number of pixels considered where both Z and Z^* are defined. We evaluate the methods considering a common subset of pixels on which all their predictions are defined. This corresponds to the area of the crop method, being the smallest.

5.2. Baseline Results

Table 1 shows the results obtained using the baseline neural architecture. It is possible to observe that our patch-based methods, especially Base-Warp, consistently outperform the baseline models across all evaluated metrics. Notably, Base-Warp achieves the highest accuracy for δ_1, while ranking second in the other δ metrics. In addition, Base-Warp obtains the lowest error values in

Table 1: Performance of the various methods w.r.t. standard MDE metrics using the baseline ResNet-based backbone.

Method	Accuracies (%) ↑			Errors ↓			
	δ_1	δ_2	δ_3	AbsRel	SqRel	RMSE	RMSLE
Base-Half	90.85	98.59	99.65	0.110	0.522	3.884	0.140
Base-Full	92.40	**99.00**	**99.77**	0.101	0.431	<u>3.396</u>	0.126
Base-Crop	<u>93.29</u>	98.85	99.71	<u>0.074</u>	<u>0.360</u>	3.417	<u>0.115</u>
Base-Warp	**93.76**	<u>98.92</u>	<u>99.73</u>	**0.071**	**0.349**	**3.350**	**0.111**

Note. Best and second-to-best scores in each column are represented as bold and underlined text.

Table 2: Performance of our methods measured in different regions of the image: left, center and right part.

Method	δ_1(%) ↑			AbsRel ↓		
	Left	Center	Right	Left	Center	Right
Base-Half	89.33	92.95	86.26	0.114	0.103	0.131
Base-Full	<u>91.35</u>	94.62	87.46	0.109	0.090	0.125
Base-Crop	90.91	<u>95.31</u>	<u>90.13</u>	<u>0.086</u>	<u>0.062</u>	<u>0.094</u>
Base-Warp	**91.61**	**95.59**	**90.59**	**0.082**	**0.059**	**0.092**

AbsRel, SqRel, RMSE, and RMSLE, highlighting its superior depth estimation performance. Among the baseline methods, Base-Full achieves the second-best results across most metrics, surpassing Base-Half, which generally performs less effectively. Moreover, our proposed methods provide significant improvements over Base-Full, especially in error metrics such as AbsRel and SqRel, where they reduce both relative and squared relative errors. Furthermore, warp-extraction proves to be consistently better than crop-extraction.

Since perspective distortions less affect the center of the image, we also evaluate performance on different image sub-regions and report the results in Table 2. Specifically, we divide the full image into three equal regions: the left, center, and right part. The results show that Base-Warp performs better than Base-Crop also in the middle of the image, implying that camera-consistent training produces an overall better model. Interestingly, performance measurements for the left and right regions are asymmetric, with better metrics on the left, regardless of the method. This asymmetry likely arises because the dataset is collected using a right-hand drive vehicle, and no horizontal flipping is applied during evaluation, potentially leading to a biased scene composition.

Fig. 5 presents qualitative examples of the depth maps generated by our methods. As shown, the depth maps produced by our approaches contain more detail compared to those from the baseline methods, especially in distant regions. Additionally, Base-Crop and Base-Warp exhibit variations in the shape of their depth prediction regions, reflecting the distinct patch-based extraction and re-projection strategies employed by each method. The shape of these pre-

Table 3: Performance of the various methods w.r.t. standard MDE metrics using the baseline ResNet-based backbone trained on Cityscapes.

Method	Accuracies (%) ↑			Errors ↓			
	δ_1	δ_2	δ_3	AbsRel	SqRel	RMSE	RMSLE
Cityscapes							
Base-Full	91.44	97.99	99.31	0.091	0.868	5.923	0.149
Base-Crop	91.61	98.01	**99.36**	0.085	**0.788**	**5.867**	**0.144**
Base-Warp	**91.62**	**98.10**	**99.36**	**0.082**	0.794	5.909	0.145

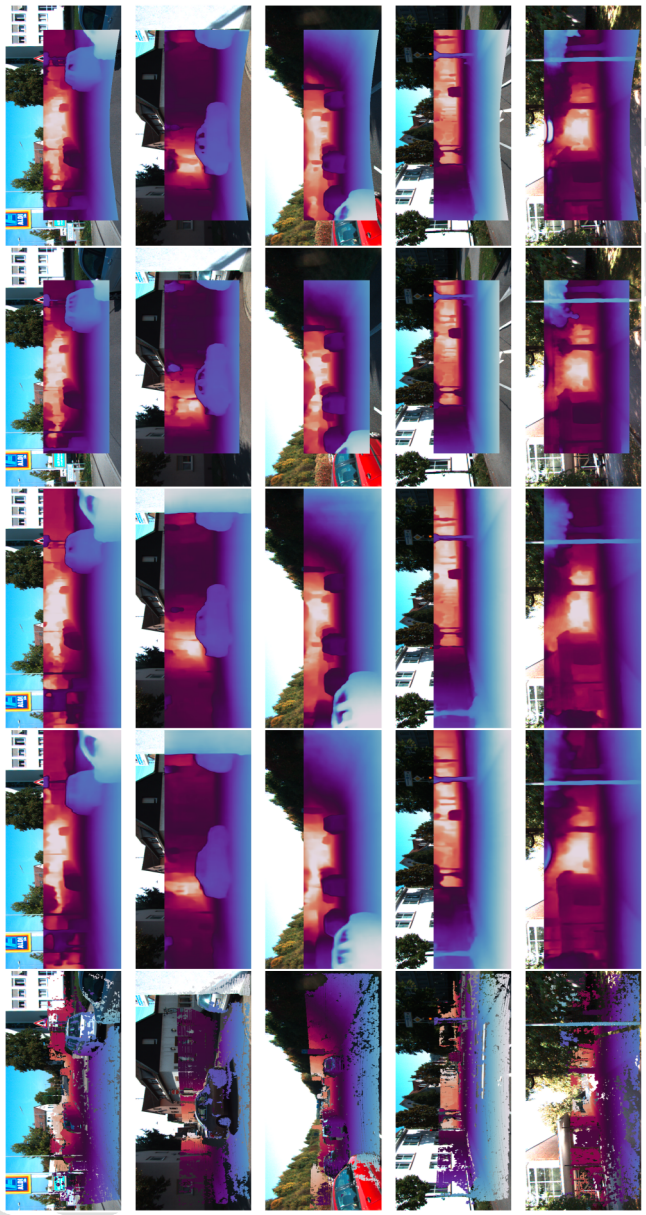
dictions can be controlled through the patch selection process, as detailed in Section 4.1. It is also worth noting that all methods inaccurately predict the depth values for the top-center region of the images, as no ground truth depth values are available for the sky regions during training.

To study the validity of the method for more varied datasets, we conducted experiments using the baseline ResNet50-based backbone on additional standard Monocular Depth Estimation (MDE) benchmarks: Cityscapes [35] (outdoor, driving, different intrinsics) and NYU Depth V2 [36](indoor, small field-of-view).

Cityscapes. is a driving scenes dataset providing about 40K densely labeled samples for MDE. Its images are characterized by an height, a width and a focal length of respectively 1024 px, 2048 px, and 2200 mm; while KITTI parameters are roughly 370 px, 1240 px, and 720 mm. Hence, CS samples have a narrower field of view, providing smaller patch datasets. We consider depth values over 100 m as not valid. For the full-image case, we trained a model on the dataset images halved in resolution due to memory constraints. The same pre-processing was applied for the crop. The warp approach automatically handles these transformations as we specified a focal length of 720 mm, for continuity with the KITTI dataset. Patches have a shape of 400×600 px. Cityscapes presents a new challenge with a significantly higher focal length and different aspect ratio than KITTI (1024×2048 px, $f \approx$2200 mm vs. KITTI's 370×1240 px, $f \approx$720 mm).

As shown in Table 3, the Base-Warp method outperforms Base-Crop and Base-Full in the most representative metrics, achieving the best AbsRel (0.082), δ_1 (91.62%), and δ_2 (98.10%), demonstrating a clear advantage on a large-scale outdoor dataset with varied camera parameters.

NYUv2. is a standard benchmark for indoor MDE. We use the NYUv2 pre-processed version by Lee et al. [37], counting 23k training samples and 654 test images. As validation data we use 5% of the training set. Images have an height of 480 pixels and a width of 640 pixels, but they contain a white border due to the applied pre-processing, so 16 pixels are cropped out from each side leading to a final shape of 448×608. The focal length is about 520mm.



(a) Ground-truth (b) Base-Half (c) Base-Full (d) Base-Crop (e) Base-Warp

Figure 5: Qualitative comparison of the various analyzed methods. The upper part of each prediction is cut, as no ground truth is associated with it during training. It is possible to observe how the depth maps produced by our approaches contain more detail compared to those from the baseline methods, especially in distant regions.

Hence, the resulting field-of-view is 60 degrees. While the fov is wider than the one from Cityscapes, the maximum depth value for the represented scenes is 10 meters. So the "context" required to perform the task is expected to span a larger portion of the image as closer objects occupy more image area. We train the patch models using squared patches of size 500×500 px, ensuring a good context coverage but sufficient left-out pixels to justify a patch-based approach.

As detailed in Table 4, the Base-Warp method maintains its competitive edge, proving its robustness across scene types. In fact, Base-Warp achieves the best performance across all key error metrics: AbsRel (0.131), SqRel (0.084), and RMSLE (0.165), significantly outperforming the traditional Base-Crop method. It also achieves the best δ_2 score (97.16%).

These results on datasets with drastically different scene content (indoor/outdoor) and camera intrinsics (focal length and resolution) strongly validate the core motivation of our approach: enforcing a canonical perspective projection via warping ensures better depth prediction generalizability, regardless of the input image's original camera parameters.

Corss-Dataset Evaluation. We conducted a cross-dataset experiment, training our baseline models on the KITTI dataset with patches of shape 400×600 px and evaluating them on the Cityscapes test set. This setting is particularly challenging due to gaps in scene content and camera parameters.

We report the relative performance difference of Base-Crop and Base-Warp with respect to the Base-Full model in Table 5.

While cross-dataset generalization is inherently difficult and results in a mixed quantitative outcome, a detailed analysis reveals that our Warp approach exhibits the maximum relative improvement over all the metrics in correspondence of δ_1 (+0.092).

The observed degradation in metrics like RMSE (Root Mean Square Error) for Base-Warp can be primarily attributed to the scene content domain gap (e.g., semantic and texture differences), which impacts all methods, including Base-Crop. The Base-Crop method's slightly better RMSE is likely due to the different ways geometric and content-based errors interact. However, the AbsRel and δ_1 improvements for Base-Warp highlight its strength in maintaining relative depth scale and absolute correctness even across drastically different camera systems.

5.3. Experiments with other backbones

Although there are patch-based approaches similar to our method [9] [11], they consider *affine* depth estimation for high-resolution or 360° images. We do not compare with them because our setting is *metric* depth estimation for large field-of-view images. Instead, in this subsection, we see how the proposed patch-based approach improves the performance of other metric depth estimation models. In all the evaluations, we report results that we obtained through our own testing pipeline, which is slightly different from the standard one as we consider all the available pixels for testing, since we are interested in the behavior of the models across the whole image. Hence, we allow for padding,

Table 4: Performance of the various methods w.r.t. standard MDE metrics using the baseline ResNet-based backbone trained on NYUv2.

Method	Accuracies (%) ↑			Errors ↓			
	δ_1	δ_2	δ_3	AbsRel	SqRel	RMSE	RMSLE
NYUv2							
Base-Full	83.11	97.06	**99.36**	0.133	0.086	0.466	0.168
Base-Crop	**83.80**	96.89	99.21	0.133	0.087	**0.458**	0.166
Base-Warp	83.50	**97.16**	99.32	**0.131**	**0.084**	0.465	**0.165**

Table 5: Relative difference for the metrics of Base-Crop and Base-Warp with respect to Base-Full metrics. The methods are trained on KITTI, the evaluation is performed on Cityscapes test set. Best improvements are highlighted in bold text.

Method	Accuracies ↑			Errors ↓			
	δ_1	δ_2	δ_3	AbsRel	SqRel	RMSE	RMSLE
KITTI → Cityscapes							
Base-Crop	-0.069	**+0.012**	**+0.013**	+0.029	**-0.064**	**-0.031**	**-0.064**
Base-Warp	**+0.092**	-0.007	+0.001	**-0.049**	-0.006	+0.022	-0.042

easing the image coverage and experiments. In addition, the training procedures do not exactly match those in the respective papers, as we use the same loss function and augmentations for all models. Our results must then be interpreted as *relative*, and not absolute.

ViT. To test how our approach behaves with different architectures, we train from scratch three ViT-based Dense Prediction Transformer [26] models respectively on full-images, cropped, and warped patches using the same setting of the baseline experiments, i.e. on patches of size 300×300 pixels, focal length of 720.0. Quantitative results are shown in Table 6. The proposed methods (Crop and Warp) still prove to be beneficial in this new architectural setting, confirming the effectiveness of a patch-based approach. However, the baseline model, based on ResNet architecture, outperforms the ViT-based model, as seen in Table 1, which is expected since transformers generally require more data than fully convolutional networks [38].

NeWCRFs. Similarly to the previous experiment, we train the NeWCRFs [33] model from scratch on full images, cropped and warped patches. As input, the architecture expects an image of height and width multiples of 32, and it outputs a depth map at a quarter of the input resolution, which is eventually rescaled to match the original size. During inference, we sample five patches of shape 384×576 pixels at a focal length of 720.0, with 20% padding margin relative to the width of the patch. The results are reported in Table 7, which shows that working with patches is beneficial, although in this case Warp behaves

Table 6: Performance of ViT-based DensePredictionTransformer [26] trained from scratch on the KITTI dataset using various methods.

Method	Accuracies (%) ↑			Errors ↓			
	δ_1	δ_2	δ_3	AbsRel	SqRel	RMSE	RMSLE
Full	91.79	98.81	99.78	0.105	0.429	3.480	0.129
Crop	**93.55**	**98.95**	99.78	0.074	**0.354**	**3.423**	**0.113**
Warp	93.40	98.93	**99.79**	**0.073**	0.358	3.450	**0.113**

Table 7: Performance of NeWCRFs [33] architecture trained from scratch.

Method	Accuracies (%) ↑			Errors ↓			
	δ_1	δ_2	δ_3	AbsRel	SqRel	RMSE	RMSLE
Full	95.84	99.43	**99.90**	0.072	0.249	2.794	0.102
Crop	96.10	99.48	**99.90**	**0.060**	**0.221**	**2.623**	**0.092**
Warp	**96.12**	**99.49**	**99.90**	**0.060**	0.223	2.652	0.093

slightly worse than Crop. Qualitative results for this model are illustrated in the attached appendix. It is important to note that aesthetically, the depth maps produced by the Base-Warp and Base-Crop NeWCRFs models appear very similar. This visual resemblance supports the idea that the low-resolution output of NeWCRFs masks the geometric differences that our Warp method introduces. The slight quantitative degradation in RMSE is likely due to the increased sensitivity to a few large errors (as per our hypothesis on far-away pixels), rather than a general visual failure.

5.4. Behavior with foundation models

To demonstrate the general applicability and robustness of our proposed Warp method, we integrated it into two distinct and highly competitive State-of-the-Art (SOTA) monocular depth estimation frameworks: Metric3Dv2 [4] and UniDepthv2 [1].

Metric3D v2. We also consider pre-trained Metric3D v2 [4] with the large ViT backbone for our experiments. The input image must be preprocessed to satisfy two requirements: a size 616×1064 pixels, and captured with a 1000.0 focal length pinhole camera. To accomplish these, we apply resize and padding as described in [4]. In particular, in the resizing process, we keep track of depth rescaling in order to bring the final prediction back to the original scale. Hence, since the required aspect ratio is ≈ 1.73, we sample 5 patches of height 288 pixels, width $288 \times 1.73 \approx 498$ pixels, with a focal length of 720.0. This size ensures that no padding is applied to the lateral patches during the pre-processing performed by the model or after the warp. The model does not improve using our

Table 8: Performance of Metric3d v2 [4]. The model is tested both as it is and fine-tuned.

Method	Accuracies (%) ↑			Errors ↓			
	δ_1	δ_2	δ_3	AbsRel	SqRel	RMSE	RMSLE
No tuning							
Full	**97.22**	99.49	99.86	**0.054**	**0.219**	2.528	**0.085**
Crop	96.72	**99.65**	**99.90**	0.071	0.233	**2.368**	0.097
Warp	96.18	99.59	99.89	0.074	0.238	2.401	0.101
Fine-tuned							
Full	98.29	99.77	99.94	0.047	0.141	2.099	0.072
Crop	98.31	**99.84**	99.95	0.048	0.123	1.876	0.069
Warp	**98.42**	99.83	**99.96**	**0.046**	**0.113**	**1.840**	**0.068**

patch-based approach, be it Crop or Warp. This is expected because the authors trained the model by aligning only the focal length and depth scale of the various images through the canonical camera transformation [4] and applying a patch-based approach (even our Warp method) introduces a data distribution shift, making fine-tuning on the new patch dataset necessary for successful generalization. However, after a fine-tuning of 5 epochs for all the methods, Warp surpasses both Crop and Full in almost all the metrics. Quantitative results are shown in Table 8 and confirm the efficacy of extracting patches by warping.

UniDepth v2. UniDepthV2 [1] is a foundation model for depth estimation. Unlike Metric3D v2, it can infer camera parameters and, if provided, have them as additional input. It accepts images with an aspect ratio ranging in [0.5, 2.5] and applies resizing if the number of pixels to process (i.e. $H \times W$) is not in a predefined interval. Moreover, height and width of the input image must be a multiple of 14 pixels. To evaluate our approach with UniDepthV2, we sample 3 patches of shape 294×588 pixels with a focal length of 720.0. We test the model in two ways, first by using the camera parameters as additional inputs and then by letting the model infer them. In Table 9 we can observe that the Warp method outperforms the other in both situations. Interestingly, the Crop method performs really poorly when the camera parameters are provided. The reason is that, when the model was trained, the authors applied only mild relative translations to the image [1]. According to Eq. 7, patches that are sampled at the borders of the image present extreme values for the coordinates of the principal point, resulting in outliers for the training distribution. This is also confirmed by the superiority of Warp in the setting without camera, as for the models it is arguably easier to infer camera parameters. The strong performance of UniDepthv2 is explained by its spherical parametrization, which inherently removes perspective distortion and makes cropped patches structurally similar to warped patches. Furthermore, UniDepthv2 estimates distance (which is equivariant to camera rotation around the center, unlike depth), minimizing the

Table 9: Performance of UniDepth v2 [1]. Two scenarios are tested. w/o camera: the model has to infer camera paramers; w/camera: the model explicitly receives the camera parameters as input.

Method	Accuracies (%) ↑			Errors ↓			
	δ_1	δ_2	δ_3	AbsRel	SqRel	RMSE	RMSLE
w/o camera							
Full	**92.56**	**99.07**	99.62	0.128	0.616	3.653	**0.142**
Crop	87.60	98.81	99.69	**0.113**	0.566	3.342	**0.142**
Warp	88.27	99.02	**99.70**	0.121	**0.493**	**3.323**	0.153
w/camera							
Full	96.45	99.22	99.70	0.073	0.349	2.891	0.107
Crop	43.95	84.88	99.29	0.214	1.122	4.985	0.307
Warp	**96.94**	**99.31**	**99.74**	**0.062**	**0.286**	**2.629**	**0.096**

error introduced during our final depth re-projection step. These factors make it naturally robust to patch-based methods.

5.5. Ablation Experiments

The most important hyper-parameter of our method is the *patch size*, together with the focal length it determines the training dataset for the depth estimation model. However, the focal length main role is to scale the image plane. In general, to avoid producing scaling artifacts during the patch extraction procedure, it is better to keep the focal length fixed. However, in case of cross-dataset validation a different focal length from the image native one can be chosen. For the KITTI dataset, the focal length is always fixed at 720 in this work. All the measurements from this subsection are performed on the KITTI validation set.

Patch Height. Different patch sizes provide different context to the neural depth predictor. To understand the behavior of the model in relation to the patch size independently of the extraction method, we train baseline models on patches of varying width and height. Specifically, we use 20% of the training subset for 10 epochs, and evaluate them on 50% of the validation subset, using the full inference pipeline. Each model is trained on patches of a given shape, extracted using either the crop-based or warp-based technique. Performance results are averaged over the two extraction methods, as they exhibit similar trends and for ease of visualization. At inference time, patches are extracted in the same manner as during training. The model architecture trained in all the experiments from this subsection is the one based on ResNet50 from [24].

The evaluation results are presented in Fig. 6. From the figure, it is possible to observe that vertical context proves to be critical for successful depth estimation, as demonstrated by the steep slope of the varying-patch-height curve.

Our findings are consistent with those of Dijk and Croon [30], further suggesting that vertical context is necessary for MDE. Most notably, the graph reveals that horizontal context reaches a nearly optimal value of accuracy around a patch width of 600pixels. Beyond this width, training on wider patches does not result in significant performance improvements and may even degrade accuracy. This further supports the effectiveness of our patch-based approach.

Patch Width. To validate the actual effect of cropping or warping on the quality of the patch dataset and the impact of out-of-boundary pixels, we train the model on patches of fixed height of 400 pixels. We Consider various widths and for each resulting patch size two models are trained: one on the cropped patches and one on the warped patches. This time we use all the training and validation sets for 10 epochs.

Results are plotted in Figure 7. Patch locations are defined by a grid built from the leftmost *admissible* location to the rightmost *admissible* location. For different patch widths the admissible locations change, since bigger patches need to be sampled closer to the image center to avoid large padded regions. As expected, predictions are more accurate for the model trained on warped patches, in particular close the image boundaries where perspective distortions are more present. Around the image center, the two approaches are on-par, with a slight advantage for the model trained on warped patches. Note that the performance around the center is roughly the same for all patch widths, while the behavior close to the left and right image margins drives the overall mean accuracy. As precised above, patch location spacing decreases with patch width and the admissible locations shrink towards the center. This implies that the leftmost and rightmost patches of width 800 pixels present more context than their corresponding patches large 200 pixels, raising performance.

Number of Patches. We study the impact of the number of patches on the method performance. Patches are sampled from a grid of equally spaced centers, left and right locations are fixed respectively to the leftmost and rightmost admissible patch locations in the image. Patches have shape 400×600, at least three are considered so to cover the whole image. Results are shown in Figure 8. We can see that the error tends to plateau with the number of patches increasing. The reason for that is the increasing overlap between patches, resulting in many pixels associated to more than one prediction each. This acts as an "ensemble" of predictors for each pixel location, generally reducing the error due to averaging effects. However, as it will be discussed in subsection 5.7, increasing the number of patches have a big impact on the overall computational cost.

5.6. Scaling Properties

We investigate how the performance of our method scales with the number of parameters. Following the procedure outlined in 5.1, we train and test baseline methods with progressively larger backbones (ResNet-18, ResNet-34, and ResNet-50) and compare their performance. The results are shown in Fig. 9,

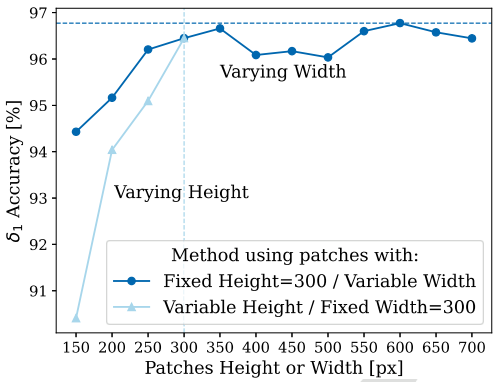


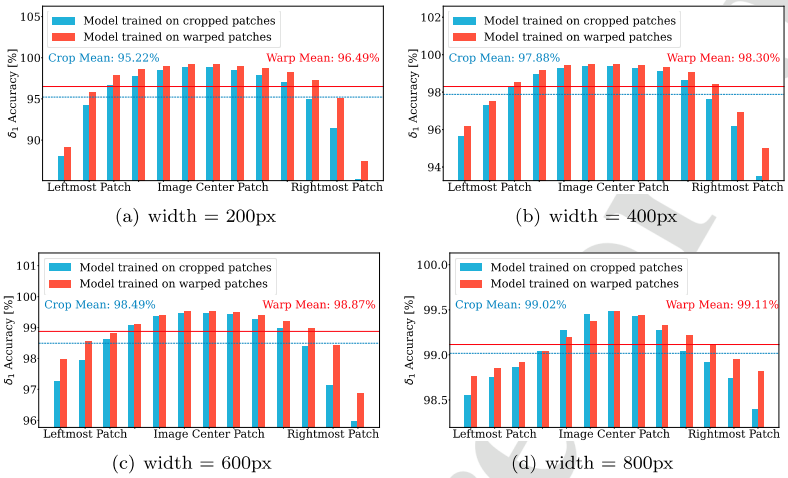
Figure 6: Validation of patch aspect ratio. The plot illustrates the average performance of our crop-based and warp-based methods, trained and evaluated using the reported patch shapes. The vertical line marks the maximum patch height we tried for the experiment, while the horizontal line is the highest accuracy value reached.

demonstrating that the four models exhibit different scaling behaviors. Specifically, Base-Half benefits the most from model size scaling, with a relative improvement of up to 25% (see Fig. 9(b)), while Base-Full shows only modest improvements with larger backbones. In contrast, our Base-Crop and Base-Warp scale more effectively than Base-Full. In particular, the performance of Base-Warp with ResNet-18 is comparable to that of Base-Full with ResNet-50, as demonstrated by the horizontal lines in the graphs. This indicates that our method not only scales better than models that process the entire image but also requires smaller networks to achieve competitive performance.

5.7. Computational Cost

Our inference pipeline employs various image transformations in addition to performing the depth estimation model forward pass. However, the patch prediction step processes smaller tensors than the full image. We study here the computational cost of the whole procedure.

Time Complexity. For a given depth estimation model ϕ, call $\phi(H, W)$ the time computational complexity of processing a tensor of shape $H \times W$. Let $P_{h,w}(H, W)$ represent the time computational complexity of projecting an image of shape $h \times w$ to an image of shape $H \times W$ through a warp transformation. Then, the overall cost $C_{H,W}(n, h, w)$ of processing an image of shape $H \times W$ using our approach with n patches of size $h \times w$ is the following:



(a) width = 200px (b) width = 400px

(c) width = 600px (d) width = 800px

Figure 7: Average performance across image patches for varying widths. Height is fixed to 400 pixels and 13 patches are considered. Each model is evaluated on patches extracted with the same patch extraction method they were trained on. Patch location spacing decreases with patch width. Mean performance values are represented as horizontal lines.

$$\begin{aligned} C_{H,W}(n,h,w) &= \sum_{i=1}^{n} \big(P_{H,W}(h,w) + \phi(h,w) + 2P_{h,w}(H,W)\big) + O(nHW) \\ &= n\,\big(P_{H,W}(h,w) + \phi(h,w) + 2P_{h,w}(H,W) + O(HW)\big) \end{aligned} \tag{18}$$

Notice that the term $P_{h,w}(H,W)$ is counted twice because the re-projection of the fusion weights is required. The last $O(nHW)$ is the time complexity of the fusion step. For continuity in the implementation, we apply a perspective transform to perform patch crops. Hence, in this subsection we don't make a distinction between crop and warp approaches.

To compare the computational cost of our method with the Full-image processing cost $\phi(H,W)$, we rewrite Eq. 18 as an overhead L plus the model forward time:

$$C_{H,W}(n,h,w) = L(n,h,w,H,W) + n\,\phi(h,w) \tag{19}$$

L is always linear in its arguments, while the behavior of $\phi(h,w)$ depends on the chosen neural network. If a fully-convolutional model is employed (e.g. a ResNet-based one), then ϕ is linear in its arguments and we have:

$$C_{H,W}(\frac{W}{w},H,w) \approx \frac{W}{w}\big(\phi(H,w) + \text{overhead}\big) > \frac{W}{w}\phi(H,w) \approx \phi(H,W) \tag{20}$$

If the whole image needs to be covered, the inference pipeline is always slower than the plain model. Nonetheless, our method can speed up inference

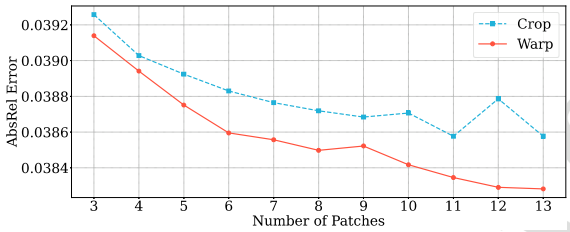


Figure 8: Absolute Relative error of the full pipeline for two ResNet50 models trained on cropped and warped patches of 400 × 600 px. The whole image is fully covered for every number of patches depicted.

Table 10: Inference time and flops for various backbones. Full-image inference and minimum patch coverage ($n = \lceil W/w \rceil + 1$) are compared on KITTI images.

Backbone	Full-Image		Patch (min coverage)	
	FPS	FLOPs (B)	FPS	FLOPs (B)
ResNet50 [24] 59.77M params	**32.40**	**554**	28.40	554
ViT [26] 123.15M params	11.7	839	**12.90**	**766**

for neural networks whose cost $\phi(h, w)$ is quadratic in w, like a DPT [26] and, more generally, ViT-based architectures.

In Table 10 we report inference speed measurements of our pipeline for various choices of the depth prediction model. Experiments refer to a DellPrecision7680 laptop mounting a NVIDIA RTX 3500 Ada, 12 GB GDDR6. As expected, convolutional based networks are always slowed down by the overhead, while ViT-based models are sped up in the minimum coverage configuration (i.e. $n \approx W/w$).

6. Conclusions

In this paper, we proposed a novel patch-based approach for metric Monocular Depth Estimation (MDE) on wide-aspect-ratio images, a common scenario in autonomous driving and robotic applications. Our method employs a trained MDE patch-model within an inference pipeline that first decomposes the input image into fixed-size patches before prediction, then merges the partial outputs to generate the final depth estimation. We extract patches either preserving camera-parameters through a perspective transformation or using a simple crop. The results demonstrate that our patch-based methodology outperforms architectures trained on full images and highlights the benefits of camera-consistent warping over simple cropping techniques, leading to more accurate depth maps. Furthermore, our studies indicates that accurate monocular depth estimation does not require the full field of view typically exploited by existing approaches that process entire images, and our pipeline improves current architectures by

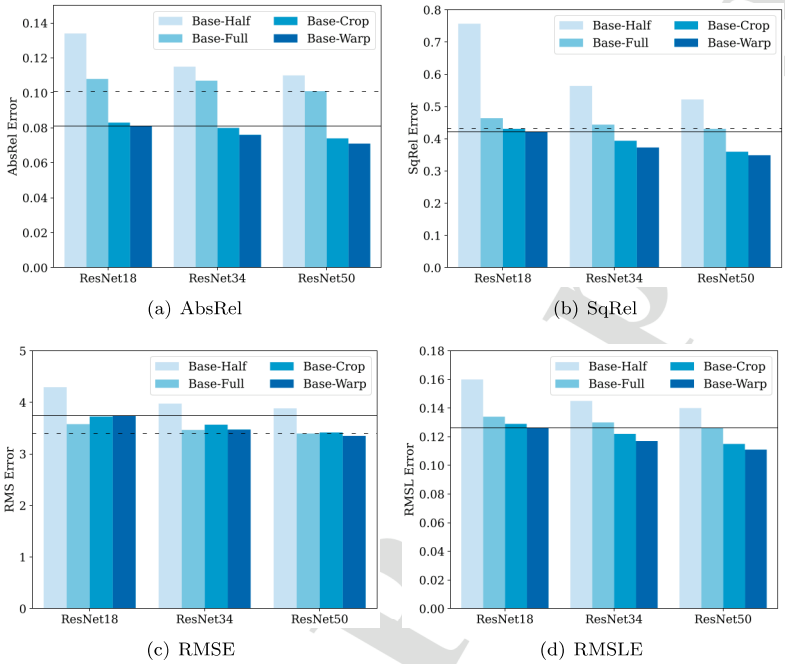


Figure 9: Performance scaling as a function of backbone size for different error metrics. The horizontal dashed line represents the performance of the Base-Full method with a ResNet-50 backbone, while the solid line corresponds to the Patch-MDE (Warp) method using a ResNet-18.

training and applying them on patches. Future work will explore adaptive patch-sizing strategies and training procedures that jointly optimize a neural patch selector alongside the patch-based MDE model.

Acknowledgments

This work was supported in part by the EC under projects EdgeAI (101097300) and by project SERICS (PE00000014) under the MUR NRRP funded by the EU – NGEU. We also thank the NVIDIA Corporation for the GPU donated. Project EdgeAI is supported by the Chips Joint Undertaking and its members including top-up funding by Austria, Belgium, France, Greece, Italy, Latvia, Netherlands, and Norway under grant agreement No 101097300. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union, the Chips Joint Undertaking, or the Italian MUR. Neither the European Union, nor the granting authority, nor Italian MUR can be held responsible for them.

References

[1] L. Piccinelli, C. Sakaridis, Y.-H. Yang, M. Segu, S. Li, W. Abbeloos, L. V. Gool, UniDepthV2: Universal monocular metric depth estimation made simpler (2025).

[2] A. Moreau, M. Mancas, T. Dutoit, Depth prediction from 2d images: A taxonomy and an evaluation study, Image and Vision Computing 93 (2020).

[3] S. F. Bhat, R. Birkl, D. Wofk, P. Wonka, M. Müller, Zoedepth: Zero-shot transfer by combining relative and metric depth, arXiv preprint arXiv:2302.12288 (2023).

[4] M. Hu, W. Yin, C. Zhang, Z. Cai, X. Long, H. Chen, K. Wang, G. Yu, C. Shen, S. Shen, Metric3d v2: A versatile monocular geometric foundation model for zero-shot metric depth and surface normal estimation, IEEE Transactions on Pattern Analysis and Machine Intelligence (2024).

[5] L. Yang, B. Kang, Z. Huang, Z. Zhao, X. Xu, J. Feng, H. Zhao, Depth anything v2, arXiv:2406.09414 (2024).

[6] Z. Li, S. F. Bhat, P. Wonka, Patchfusion: An end-to-end tile-based framework for high-resolution monocular metric depth estimation, in: 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024, pp. 10016–10025.

[7] Z. Li, S. F. Bhat, P. Wonka, Patchrefiner: Leveraging synthetic data for real-domain high-resolution monocular metric depth estimation, in: Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LXVII, Springer-Verlag, Berlin, Heidelberg, 2024, p. 250–267.

[8] J. Yan, H. Zhao, P. Bu, Y. Jin, Channel-wise attention-based network for self-supervised monocular depth estimation (2021).

[9] S. M. H. Miangoleh, S. Dille, L. Mai, S. Paris, Y. Aksoy, Boosting monocular depth estimation models to high-resolution via content-adaptive multi-resolution merging, in: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 9680–9689.

[10] Y. Liu, C. Chen, Mode: Monocular omnidirectional depth estimation via consistent depth fusion, Image and Vision Computing 136 (2023) 104723.

[11] M. Rey-Area, M. Yuan, C. Richardt, 360monodepth: High-resolution 360° monocular depth estimation, in: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 3752–3762.

[12] A. Saxena, M. Sun, A. Y. Ng, Make3d: Learning 3d scene structure from a single still image, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (5) (2009) 824–840.

[13] D. Hoiem, A. A. Efros, M. Hebert, Automatic photo pop-up, ACM Trans. Graph. 24 (3) (2005) 577–584.

[14] K. Karsch, C. Liu, S. B. Kang, Depth transfer: Depth extraction from video using non-parametric sampling, IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (11) (2014) 2144–2158.

[15] D. Eigen, C. Puhrsch, R. Fergus, Depth map prediction from a single image using a multi-scale deep network, in: Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, Vol. 27, Curran Associates, Inc., 2014.

[16] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, N. Navab, Deeper depth prediction with fully convolutional residual networks, in: 2016 Fourth International Conference on 3D Vision (3DV), 2016, pp. 239–248.

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

[18] Y. Cao, Z. Wu, C. Shen, Estimating depth from monocular images as classification using deep fully convolutional residual networks, IEEE Transactions on Circuits and Systems for Video Technology 28 (11) (2018) 3174–3182.

[19] H. Fu, M. Gong, C. Wang, K. Batmanghelich, D. Tao, Deep ordinal regression network for monocular depth estimation, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 2002–2011.

[20] S. Farooq Bhat, I. Alhashim, P. Wonka, Adabins: Depth estimation using adaptive bins, in: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 4008–4017.

[21] S. F. Bhat, I. Alhashim, P. Wonka, Localbins: Improving depth estimation by learning local distributions, in: S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, T. Hassner (Eds.), Computer Vision – ECCV 2022, Springer Nature Switzerland, Cham, 2022, pp. 480–496.

[22] W. Chen, Z. Fu, D. Yang, J. Deng, Single-image depth perception in the wild, in: Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16, Curran Associates Inc., Red Hook, NY, USA, 2016, pp. 730–738.

[23] K. Xian, C. Shen, Z. Cao, H. Lu, Y. Xiao, R. Li, Z. Luo, Monocular relative depth perception with web stereo data supervision, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 311–320.

[24] K. Xian, J. Zhang, O. Wang, L. Mai, Z. Lin, Z. Cao, Structure-guided ranking loss for single image depth prediction, in: The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[25] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, V. Koltun, Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer, IEEE Transactions on Pattern Analysis and Machine Intelligence 44 (3) (2022) 1623–1637.

[26] R. Ranftl, A. Bochkovskiy, V. Koltun, Vision transformers for dense prediction, in: 2021 IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 12159–12168.

[27] B. Ke, A. Obukhov, S. Huang, N. Metzger, R. C. Daudt, K. Schindler, Repurposing diffusion-based image generators for monocular depth estimation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024.

[28] Y. Li, Y. Guo, Z. Yan, X. Huang, Y. Duan, L. Ren, OmniFusion: 360 Monocular Depth Estimation via Geometry-Aware Fusion , in: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Los Alamitos, CA, USA, 2022, pp. 2791–2800.

[29] C. Zhang, W. Yin, B. Wang, G. Yu, B. FU, C. Shen, Hierarchical normalization for robust monocular depth estimation, in: S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, A. Oh (Eds.), Advances in Neural Information Processing Systems, Vol. 35, Curran Associates, Inc., 2022, pp. 14128–14139.

[30] T. Van Dijk, G. De Croon, How do neural networks see depth in single images?, in: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 2183–2191.

[31] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? the kitti vision benchmark suite, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 3354–3361.

[32] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, A. Geiger, Sparsity invariant cnns, in: 2017 International Conference on 3D Vision (3DV), 2017, pp. 11–20.

[33] W. Yuan, X. Gu, Z. Dai, S. Zhu, P. Tan, Newcrfs: Neural window fully-connected crfs for monocular depth estimation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2022.

[34] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization., in: Y. Bengio, Y. LeCun (Eds.), ICLR (Poster), 2015.

[35] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The cityscapes dataset for semantic urban scene understanding, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 3213–3223.

[36] N. Silberman, D. Hoiem, P. Kohli, R. Fergus, Indoor segmentation and support inference from rgbd images, in: A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, C. Schmid (Eds.), Computer Vision – ECCV 2012, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 746–760.

[37] J. H. Lee, M.-K. Han, D. W. Ko, I. H. Suh, From big to small: Multi-scale local planar guidance for monocular depth estimation, arXiv preprint arXiv:1907.10326 (2019).

[38] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16x16 words: Transformers for image recognition at scale, ICLR (2021).

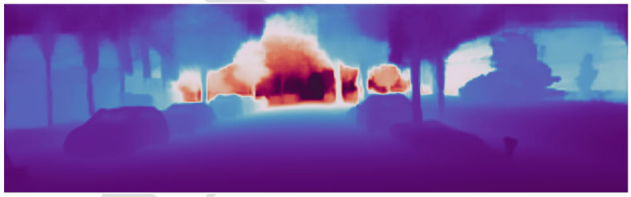
Appendix A. Qualitative results

We report some qualitative comparisons for the NeWCRF [33] model trained with our Crop and Warp method. Despite not strong numerical results, the Warp prediction is on par with the Crop method.

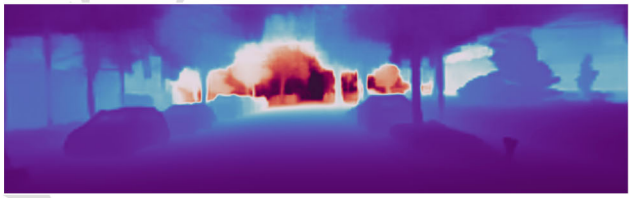
(a) Input Image and Ground Truth.

(b) Cropped Patches.

(c) Warped Patches.



(d) NeWCRF-Crop Prediction.



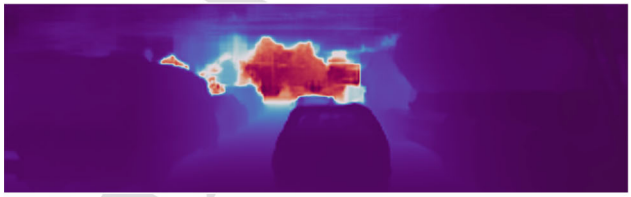
(e) NeWCRF-Warp Prediction.

Figure A.10: Example 1. Qualitative results for NeWCRF trained and employed with the proposed methods.

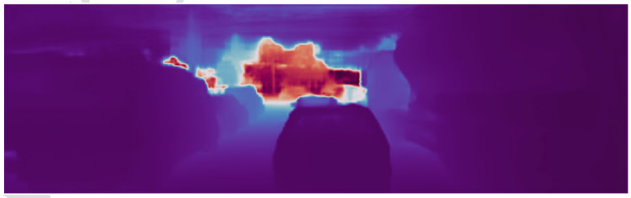
(a) Input Image and Ground Truth.

(b) Cropped Patches.

(c) Warped Patches.



(d) NeWCRF-Crop Prediction.



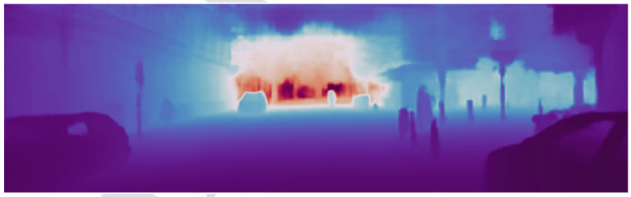
(e) NeWCRF-Warp Prediction.

Figure A.11: Example 2. Qualitative results for NeWCRF trained and employed with the proposed methods.

(a) Input Image and Ground Truth.

(b) Cropped Patches.

(c) Warped Patches.



(d) NeWCRF-Crop Prediction.

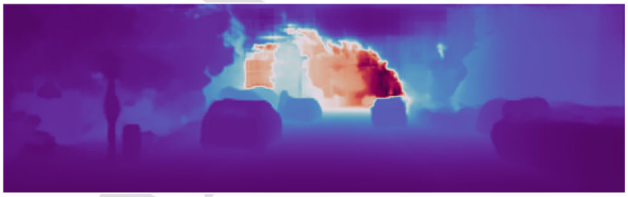
(e) NeWCRF-Warp Prediction.

Figure A.12: Example 3. Qualitative results for NeWCRF trained and employed with the proposed methods.

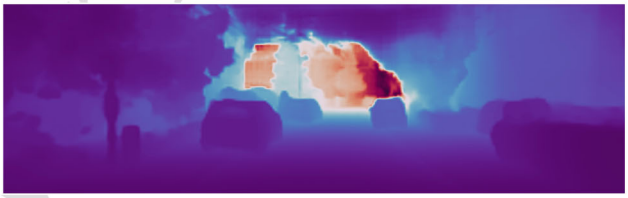
(a) Input Image and Ground Truth.

(b) Cropped Patches.

(c) Warped Patches.



(d) NeWCRF-Crop Prediction.



(e) NeWCRF-Warp Prediction.

Figure A.13: Example 4. Qualitative results for NeWCRF trained and employed with the proposed methods.