Alma Mater Studiorum · Università di Bologna

Department of Computer Science and Engineering
Master's Degree in Artificial Intelligence

Mitigating non-Lambertian surfaces issues in Stereo Matching with Neural Radiance Fields

Master Thesis
in
Computer Vision

Supervisor:
Prof. Luigi Di Stefano

Candidate:
Alex Costanzino

Co-Supervisors:
PhD. Fabio Tosi
PhD. Pierluigi Zama Ramirez

Session II
Academic Year 2021/2022

Mitigating non-Lambertian surfaces issues in Stereo Matching with Neural Radiance Fields

Abstract

Depth estimation from images has long been regarded as a preferable alternative compared to expensive and intrusive active sensors, such as LiDAR and ToF. The topic has attracted the attention of an increasingly wide audience thanks to the great amount of application domains, such as autonomous driving, robotic navigation and 3D reconstruction.

Among the various techniques employed for depth estimation, stereo matching is one of the most widespread, owing to its robustness, speed and simplicity in setup. Recent developments has been aided by the abundance of annotated stereo images, which granted to deep learning the opportunity to thrive in a research area where deep networks can reach state-of-the-art sub-pixel precision in most cases.

Despite the recent findings, stereo matching still begets many open challenges, two among them being finding pixel correspondences in presence of objects that exhibits a non-Lambertian behaviour and processing high-resolution images.

Recently, a novel dataset named Booster, which contains high-resolution stereo pairs featuring a large collection of labeled non-Lambertian objects, has been released. The work shown that training state-of-the-art deep neural network on such data improves the generalization capabilities of these networks also in presence of non-Lambertian surfaces. Regardless being a further step to tackle the aforementioned challenge, Booster includes a rather small number of annotated images, and thus cannot satisfy the intensive training requirements of deep learning.

This thesis work aims to investigate novel view synthesis techniques to augment the Booster dataset, with ultimate goal of improving stereo matching reliability in presence of high-resolution images that displays non-Lambertian surfaces.

Acknowledgements

First and foremost, I would like to express my gratitude to Professor Luigi Di Stefano for granting me the chance to work on such a stimulating and exciting project.

Furthermore, I would like to thank Fabio Tosi and Pierluigi Zama Ramirez for guiding me since the very beginning of this project.

Finally, I cannot thank enough Matteo Poggi and, once again, Fabio Tosi for all their support and the delightful company in the laboratory over the last few months.

Contents

List of Figures

Introduction

Nowadays, depth estimation from images is a preferred alternative with respect to active sensors, thanks to its favourable performance-vs-cost trade-off, shown in Figure 1.

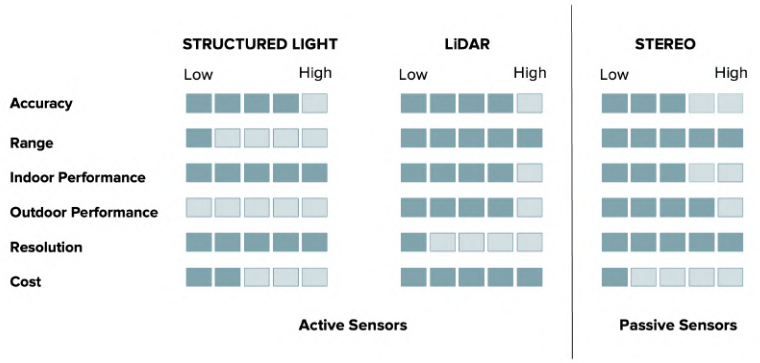


Figure 1: Performances comparison between different depth estimation technologies.

Inferring depth from an image means measuring the distance of each pixel of such image, relative to the camera that acquired it. There are several setups to estimate depth from images. Among them:

- Multi-view setup, in which multiple cameras acquire the scene from different points of view, or a single camera moves in the space while acquiring the scene;
- Two-view setup, in which a pair of cameras, usually aligned, fronto-parallel and co-planar to mimic stereoptic human vision, acquire the scene from a single point of view, with a left and right perspective;
- Mono-view setup, in which a single camera acquires the scene from a single point of view, with a single perspective.

The main focus of this thesis work is to mitigate issues that may arise in two-views setups. The focus is on this kind of setup since it has a better operability with respect to multi-view systems, with autonomous driving systems being a clear case, as depicted in Figure 2.

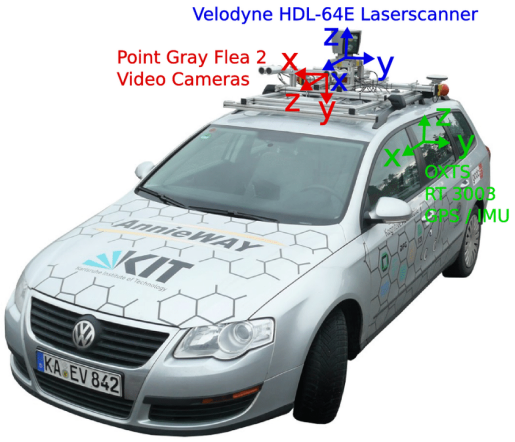


Figure 2: Two-view setups can quickly acquire a scene and, depending on the underlying algorithm, perform real-time inference.

Moreover, the two-view setup tends to be more precise with respect to the monocular depth estimation one, since the former is based on a geometric formulation.

The fundamental algorithm to infer depth from images is the so called stereo matching algorithm, that consists into finding pixels in the two images that correspond to the same 3D point in the scene, to triangulate its position. Traditional stereo algorithms have been developed following a common pipeline. With the advent of deep learning the steps of such pipeline have been reformulated as stand-alone learnable neural networks, that required intermediate supervised data, which was difficult to collect and often imprecise. Nowadays, there exist end-to-end architectures that can reach sub-pixel precision, which only require input stereo pairs with the associated ground-truth as supervision.

Despite the outstanding results, stereo matching still presents two major issues:

(i) Finding pixel correspondences in presence of non-Lambertian surfaces, since, due to inconsistencies in the left and right view, it may not be always possible to find satisfying matches;

(ii) Processing high-resolution images, since larger disparity ranges leads to a greater number of occluded and untextured pixels, in which once again it may not be always possible to find matches. Furthermore, processing high-resolution images generates computational complexity issues, in particular when deep networks are concerned.

Recently, a novel dataset named Booster, which contains high-resolution stereo pairs featuring a large collection of labeled non-Lambertian objects, has been released. The work also shown that training state-of-the-art deep neural network on such data improves the generalization capabilities of these networks, also in presence of non-Lambertian surfaces. However, since Booster is acquired with a space-time framework, followed by manual cleaning and filtering, the process to generate high-quality data can be long and cumbersome. Indeed, the Booster dataset includes a rather small number of annotated images, and thus cannot satisfy the intensive training requirements of deep learning.

To overcome this issue it is possible to extend Booster with synthetic but realistic images, generated by means of novel view synthesis techniques. Such techniques takes as input a set of sparse views of a scene, along with the respective camera poses, and generates new views of the scene, unseen during the training. At this time, one of the most widespread novel view synthesis technique is without any doubt represented by Neural Radiance Fields (NeRF). NeRF represents a scene as a continuous volumetric function, parameterized by multi-layer perceptrons, optimized to represent this mapping by regressing from a single 5D coordinate to a single volume density and view-dependent RGB color, which are later accumulated into a 2D image by means of traditional rendering techniques.

Although NeRF has demonstrated impressive results, the model presents some flaws. However, thanks to its outstanding success a long series of iterations has been proposed. Such iterations aim to:

(i) Improve scene representation capabilities by applying input encodings to the networks;

(ii) Speed-up training and inference by employing efficient hash encoding of input, output and weights or by means of knowledge distillation;

(iii) Endow the models with relighting and material editing capabilities by scene factorization.

The thesis work is organized as follows:

- Chapter 1 provides the required background on depth estimation with a literature review on stereo matching, benchmarks and novel view synthesis techniques;
- Chapter 2 presents the space-time framework along with RAFT-Stereo, a state-of-the-art stereo matching deep neural network employed to produce the Booster dataset;
- Chapter 3 depicts the main novel view synthesis techniques based on NeRF, later employed to perform experiments to augment the Booster dataset;
- Chapter 4 shows the performed experiments within the relevant results and considerations about them. Also a methodology to augment the Booster dataset with high-quality data is devised.

Chapter 1

Background and Related Works

In Computer Vision (CV) and Computer Graphics (CG), the 3D reconstruction task consists in capturing the shape and appearance of real-world objects. When such reconstruction is based on images, the fundamental question is: given that all points in the scene that fall along a ray to the pinhole are projected to the same point in the image, how is possible to recover depth information?

There is no single unified theory for scene reconstruction yet [29]. Various techniques uses disparate visual cues, such as motion, binocular stereopsis, multiple views or texture.

1.1 Depth estimation

Depth estimation is the task of measuring the distance of each pixel of an image relative to the camera [50]. Traditional methods employs multi-view geometry to find matches between images, while newer approaches directly estimate depth by minimizing a regression loss or by learning to generate a novel view from a sequence of images.

Extracting depth information from images has always been a challenging task. Due to technological constraints, such as hardware accuracy and rudimental algorithms, earlier solutions required expensive equipment as well as unavoidable human intervention, resulting in inaccuracy in the majority of situations nonetheless. Recently, as a result of technology advancements, many approaches for estimating accurate depth information from single or multiple images have been developed.

Currently there are two main methods for depth estimation based on Computer Vision [18]:

- **Active methods**, in which a controlled energy beam is transmitted towards the target and then the reflected echo is captured through a receiving device;
- **Passive methods**, in which the object is exposed to natural light and captured by a camera.

Active methods can achieve an higher accuracy at a correspondingly higher cost, also with several restrictions on the use scenario. On the other hand, passive methods have lower accuracy, but contained cost and better operability.

1.1.1 Active methods

The fundamental idea behind active methods is to generate a controlled energy beam, then, receive and process the reflected energy. A receiving and sensing device, as well as subsequent processing, are required in addition to the imaging equipment. Active methods provide are more accurate than passive methods, which use imaging equipment directly for depth information processing, but they also have the drawbacks of sophisticated technological implementation and a higher cost.

1.1.1.1 Structured Light

A Structured Light (SL) setup consists in the employment of infrared lasers to project light with certain structural characteristics onto the subject, as depicted in Figure 1.1, while an infrared camera collects and reflects the structured light pattern, extracting depth information according to the principle of triangulation [32, 11].

Such technique does not depend on the color and texture of the object itself. Actively projecting a known pattern leads to fast and robust matching of feature points, which can achieve higher accuracy. However, it also has its shortcomings, such as being difficult to use in strong light environments, short measurement distances, and being susceptible to reflections from smooth flat surfaces.

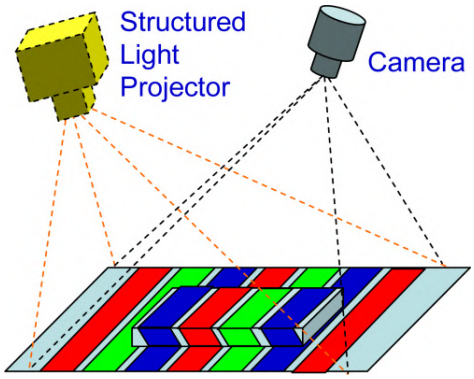


Figure 1.1: A Structured Light setup.

1.1.1.2 Time of Flight

A Time of Flight (ToF) setup works by illuminating the scene with a modulated light source and capturing the reflected light. The phase shift between the illumination and the reflection is then measured and translated to distance [6, 18]. A rudimental scheme[1] is depicted in Figure 1.2.

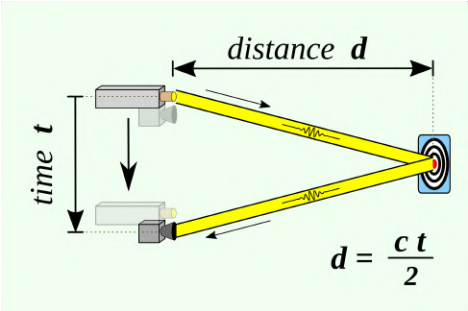


Figure 1.2: A Time of Flight setup.

Usually, the illumination is generated by a solid-state laser or a LED operating in the near-infrared range, invisible to the human eyes, while an imaging sensor, devised to operate in the same spectrum, capture the light and converts the photonic energy into electrical current. The light entering the sensor has an ambient component and a reflected component. Depth information is only embedded in the reflected component, hence, the main drawback of ToF sensors is that an high ambient component may reduces the signal-to-noise ratio.

[1] By RCraig09 - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=89646954.

However, in contrast to stereo vision or triangulation systems, ToF systems are very compact, since the illumination is placed just next to the lens, whereas other systems need a certain minimum baseline. Moreover, in contrast to laser scanning systems, no mechanical moving parts are needed. Besides, since the distance information is extracted directly from the output signals of the ToF sensor, the process requires little computational resources, in contrast to stereo vision, which utilizes complex algorithms. Furthermore, ToF cameras are able to measure the distances within a complete scene with a single shot.

1.1.1.3 Laser scanning

Light Detection and Ranging (LiDAR) is a sensing technique that uses light in the form of a pulsed laser to measure ranges of a scene. These light pulses generate precise 3D information about the shape of the scene and its surface characteristics. This method is based on the same principle of Time of Flight methods. However, ToF emits light signals, while LiDAR emits laser beams. Moreover, ToF signals are easier to interfere and the accuracy and speed will decrease as the distance is farther, while LiDAR laser beam has anti-interference properties and is suitable for long-distance ranging.

The functional difference between LiDAR and other kinds of ToF techniques is that LiDAR uses pulsed lasers to generate a point cloud, which is then used to construct a 3D map, while ToF methods generate depth maps using light detection, which is typically accomplished with a standard RGB camera [55]. ToF advantage over LiDAR is that the former requires less specialized equipment and can thus be used with smaller and less expensive devices, while the advantage of LiDAR stems from the ease with which a computer can process a point cloud as opposed to a depth map.

1.1.2 Stereo matching

Stereo matching is the process of acquiring two or more images and building a 3D model of the scene by finding matching pixels in the images and converting their 2D positions into 3D depths [42].

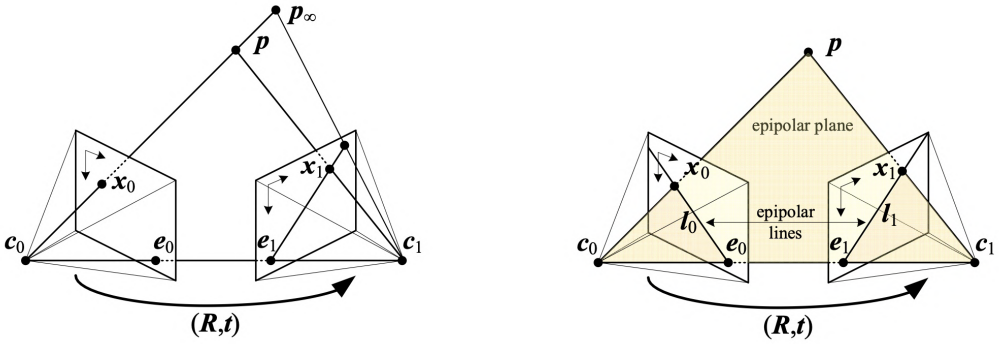
There exists a variety of search techniques that can be used to match pixels, either based on their local appearance as well as the appearance of neighbouring pixels. In the case of stereo matching there are some additional information available, namely, the positions and calibration data for the cameras that took the pictures of the same static scene. Such information can be

exploited to reduce the number of potential correspondences, speeding up the matching and increasing its reliability.

1.1.2.1 Epipolar geometry

A pixel in one image x_0 projects to an *epipolar line segment* in the other image, as shown is Figure 1.3a. Such line segment is bounded (i) at one end by the projection of the original viewing ray at infinity p_∞; (ii) at the other end by the projection of the original camera center c_0 into the second camera, which is known as the epipole e_1.

If the epipolar line in the second image is projected back into the first, another line is obtained, this time bounded by the other corresponding epipole e_0. Extending both line segments to infinity, a pair of corresponding epipolar lines are obtained, as depicted in Figure 1.3b. Such epipolar lines are the intersection of the two image planes with the epipolar plane that passes through both camera centers c_0 and c_1 as well as the point of interest p [42, 25].



(a) Epipolar line corresponding to one ray.

(b) Corresponding set of epipolar lines and their epipolar plane.

Figure 1.3: A scheme of an epipolar geometry setup.

The epipolar geometry for a pair of cameras is implicit in the relative pose and calibrations of the cameras [25]. Once such geometry has been computed, the epipolar line corresponding to a pixel in one image can be exploited to constrain the search for corresponding pixels in the other image. This search is usually devised by means of a general correspondence algorithm, such as optical flow, considering only locations along the epipolar line [42].

1.1.2.2 Rectification

A more efficient algorithm can be obtained by rectifying in prior the input images so that corresponding horizontal scanlines are epipolar lines [42, 19, 13], making possible to match horizontal scanlines independently.

A simple way to rectify the two images is to [10]:

(i) Rotate both cameras so that they are looking perpendicular to the line joining the camera centers c_0 and c_1;

(ii) Determine the desired twist around the optical axes that makes the up vector perpendicular to the camera center line, ensuring that corresponding epipolar lines are horizontal and that the disparity[2] for points at infinity is null;

(iii) If necessary, rescale the images to account for different focal lengths, magnifying the smaller image to avoid aliasing.

Image rectification is an equivalent alternative to perfect camera co-planarity. Even with high-precision equipment, it is usually performed since it may be impractical to maintain perfect co-planarity between cameras.

1.1.2.3 Standard rectified geometry

When additional information about the imaging process is available, for instance if the images were acquired on co-planar photographic plates, more specialized and accurate algorithms can be devised. The aforementioned setup is denominated standard rectified geometry and is employed in a lot of stereo camera setups and stereo algorithms, leading to a simple inverse relationship between 3D depths z and disparities d,

$$d = f\frac{b}{z} \tag{1.1}$$

where f is the focal length measured in pixels and b is the baseline [42, 13].

The equations

$$x^{'} = x + d(x, y), \quad y^{'} = y \tag{1.2}$$

[2]The concept of disparity was first introduced in the context of human vision to describe the difference in location of corresponding features perceived by the eyes.

describe the relationship between corresponding pixel coordinates in the left and right images [42]. In this manner, the task of extracting depth from a set of images becomes estimating the disparity map $d(x, y)$.

After rectification, it is easy to compare the similarity of pixels at corresponding locations (x, y) and $(x', y') = (x + d, y)$, derived from Equation 1.1 and Equation 1.2, to store them in a disparity space image (DSI) $C(x, y, d)$ for further processing [42].

1.1.2.4 Sparse correspondence

Earlier stereo matching algorithms were feature-based: at first a set of potentially matchable locations were extracted by means of interest operators, then a search for corresponding location was performed by means of window-based metrics.

Such limitation was due to: (i) computational resource limitations; (ii) necessity to limit the response of stereo algorithm to obtain more reliable matches.

Recent works focus on the extraction of highly reliable features to use them as seed to grow additional matches or as input to dense depth solvers. Similar approaches have also been employed to wide-baseline multi-view problems for 3D surface reconstruction [42].

1.1.2.5 Dense correspondence

Nowadays, sparse matching algorithms are seldom employed since the focus is directed towards dense correspondence, better suited for applications such as image-based rendering and modelling [42].

These algorithms have been developed following a common pipeline that involves: (i) matching cost computation and aggregation; (ii) disparity computation and optimization; (iii) disparity refinement [31, 42].

It is possible to distinguish two main categories [31, 42]:

- **Local methods**, in which disparity computation at a given point depends only on intensity values within a finite window. These algorithms usually make implicit smoothness assumptions by aggregating support.
- **Global methods**, which make explicit smoothness assumptions and then solve a global optimization problem. These algorithms normally

do not perform an aggregation step, since they rather seek for a disparity assignment that minimizes a global cost function consisting of data and smoothness terms.

The main difference among these algorithms is the minimization procedure adopted. Between these two classes, there are certain iterative algorithms that do not explicitly specify a global function to minimize, but whose behavior mimics closely the ones of iterative optimization algorithms, such as hierarchical coarse-to-fine algorithm, which operate on an image pyramid where results from coarser levels are used to constrain a more local search at finer levels [53]. Situated between local and global methods there is also the famous Semi-Global matching (SGM), which approximates a 2D cost function via 1D optimization [39].

1.1.2.6 Similarity measures

The first step of any dense stereo matching algorithm is a similarity measure that compares pixel values in order to determine how likely they are to be in correspondence. The use of similarity measures is quite widespread in Computer Vision, hence, a lot of functions have been devised; however, few others have been developed specifically for stereo matching [31].

Given a patch from the left image $I_L(x)$ sampled at discrete pixel locations $x_i = (x_i, y_i)$, we wish to find where it is located in right image $I_R(x)$, sliding the patch of the left image on the right image.

The most common pixel-based matching costs include:

- **Sum of Squared Differences** (SSD), defined as

$$\mathrm{SSD}(u) = \sum_i \big[I_R(x_i + u) - I_L(x_i)\big]^2$$

  where $u = (u, v)$ is the pixel displacement;

- **Sum of Absolute Differences** (SAD), defined as

$$\mathrm{SAD}(u) = \sum_i \big|I_R(x_i + u) - I_L(x_i)\big|$$

There is also a class of function based on the cross-correlation between patches, that includes:

- **Cross-Correlation** (CC), defined as

$$\mathrm{CC}(u) = \sum_i I_R(x_i + u) \cdot I_L(x_i)$$

- **Normalized Cross-Correlation** (NCC), defined as

$$\mathrm{NCC}(u) = \frac{\sum_i I_R(x_i + u) \cdot I_L(x_i)}{\sqrt{\sum_i I_R(x_i + u)^2} \cdot \sqrt{\sum_i I_L(x_i)^2}}$$

- **Zero-Mean Normalized Cross-Correlation** (ZNCC), defined as

$$\mathrm{ZNCC}(u) = \frac{\sum_i \big[I_R(x_i + u) - \mu(I_R)\big] \cdot \big[I_L(x_i) - \mu(I_L)\big]}{\sqrt{\sum_i \big[I_R(x_i + u) - \mu(I_R)\big]^2} \cdot \sqrt{\sum_i \big[I_L(x_i) - \mu(I_L)\big]^2}}$$

  where μ represents the mean function.

The latter turns out a similarity function very robust to intensity changes, which is a good feature if significant exposure or appearance variation between images to be matched are expected. However, such function is also computationally expensive.

Recently, robust measures, such as contamined Gaussians and truncated quadratics, have been proposed [42]. These measures are based on a general function[3],

$$\mathrm{SRD}(u) = \sum_i \rho\big(I_R(x_i + u) - I_L(x_i)\big) \tag{1.3}$$

where ρ is a robust norm, a function that grows less quickly than the quadratic penalty associated with least squares [41, 49]. These measures are useful since they limit the influence of mismatches during aggregation.

Interestingly, color information does not appear to be helpful when employed in matching costs [3], even though it is important for aggregation. When matching more than pairs of images, more sophisticated variants of similarity measures, based on photoconsistency, can be used [42].

One of the first successes of deep learning in stereo matching was the learning of matching costs functions. Such approach is the most widespread nowadays.

[3] SRD stands for Sum of Robust Differences.

1.1.2.7 Local methods

Local methods aggregate the matching cost by summing or averaging over a support region in the DSI. Such support region can be either two-dimensional at a fixed disparity, favoring fronto-parallel surfaces, or three-dimensional in xyd space, supporting slanted surfaces [42].

Aggregation with a fixed support region can be performed using 2D or 3D convolution, while in case of rectangular windows, using efficient moving average box-filters. Shiftable windows can be implemented efficiently using separable sliding min-filters. Selecting the right window is fundamental, since windows must be large enough to contain sufficient texture and yet small enough so that they do not pass over depth discontinuities [31, 42].

Among these local aggregation methods the fast variable window approach and the locally weighting approach developed consistently stood out having the best performance-vs-speed trade-off [47].

The emphasis for local methods is on matching cost computation and cost aggregation. The final disparities are easy to compute since it suffice to select the disparity associated with the lowest cost value for each pixel. As a result, for each pixel, these algorithms execute a local Winner-Take-All (WTA) optimization. A shortcoming of this approach is that the uniqueness of matches is only enforced for one image, whereas points in the other image may match various points unless cross-checking and subsequent hole filling are utilized [47, 42].

Sub-pixel refinement Most stereo correspondence algorithms compute a set of disparity estimates in a discretized space. For certain applications such as robot navigation and tracking, these approaches may be perfectly adequate. Instead, for image-based rendering, such quantized maps begets very degraded view synthesis results, causing scene appearance to be made up of many thin shearing layers. To amend, many algorithms employ a sub-pixel refinement stage after the initial discrete correspondence.

1.1.2.8 Global methods

Global methods perform some optimization steps after the disparity computation phase, often skipping the aggregation step altogether since the global smoothness constraints perform a similar function [31]. Most global methods are formulated as an energy-minimization framework, where the goal is to

find a solution d that minimizes a global energy,

$$E(d) = E_D(d) + \lambda E_S(d). \tag{1.4}$$

In the Equation 1.4 $E_D(d)$ is referred as data term and it measures how well the disparity function d adhere with the input image pair. Such energy is defined as

$$E_D(d) = \sum_{(x,y)} C\big(x, y, d(x,y)\big), \tag{1.5}$$

where C is the matching cost.

$E_S(d)$ is referred as smoothness term and it encodes the smoothness assumptions made by the algorithm. To make the optimization computationally tractable, such term is usually restricted to measure only the differences between neighboring pixels disparities,

$$E_S(d) = \sum_{(x,y)} \rho\big(d(x,y) - d(x+1,y)\big) + \rho\big(d(x,y) - d(x,y+1)\big), \tag{1.6}$$

where ρ is a monotonically increasing function of disparity difference. It is also possible to use larger neighborhoods, which can lead to better boundaries, or to use second-order smoothness terms, although such terms require more complex optimization techniques [42].

Once the global energy has been defined, various algorithms may be employed to seek for a local minima.

1.1.2.9 Semi-Global Matching

The Semi-Global Matching (SGM) method seeks to approximate a global regularized cost function by following one dimensional paths L in several directions r through the image [14]. According to the method it is sufficient to use eight or sixteen paths, as shown in Figure 1.4, to cover the entire structure of an image. Along each path, the minimum cost is calculated by means of dynamic programming

$$\begin{aligned} L_r(p,d) = C(p,d) + \min\big[&L_r(p-r,d), L_r(p-r,d-1) + P_1, \\ &L_r(p-r,d+1) + P_1, \min_i L_r(p-r,i) + P_2\big] \end{aligned}$$

For every pixel p and disparity d the cost is calculated as the sum of the matching cost $C(p,d)$ and the minimum path cost to the previous pixel,

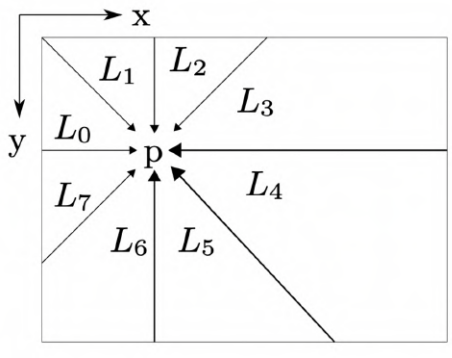


Figure 1.4: Path directions in Semi-Global matching with eight directions.

with the penalties P_1 and P_2. P_1 penalizes slanted surfaces while P_2 penalizes discontinuities [39].

The information from all paths is then summed for all pixels and disparities, building the accumulated costs defined as

$$C_a(p,d) = \sum_r L_r(p,d)$$

The disparity for each pixel is chosen by a Winner-Takes-All strategy on S. In contrast to other dynamic programming solutions, explicit occlusion handling is not possible, hence a left-right consistency check is applied, either using (i) the disparities of the right image calculated with the same process; (ii) by diagonal search in S.

1.1.3 Deep stereo matching

While early work focused on designing better matching cost formulations and efficient inference algorithms [16], with the predominant advent of deep learning, the subsequent research efforts were directed towards the reformulation of the stereo matching pipeline's steps as stand-alone learnable neural networks [56].

1.1.3.1 Semantic depth estimation

Recent developments have shown that semantic cues from image segmentation can be used to improve the results of stereo matching [52]. Semantic labels improve the accuracy of stereo matching in untextured, occluded and reflective regions, while depth information obtained by stereo matching can

be used to solve possible confusions between similar semantic categories. Thus, segmentation information and depth maps complement each other, representing high-level information of the scene. The general scheme of such networks is depicted in Figure 1.5.

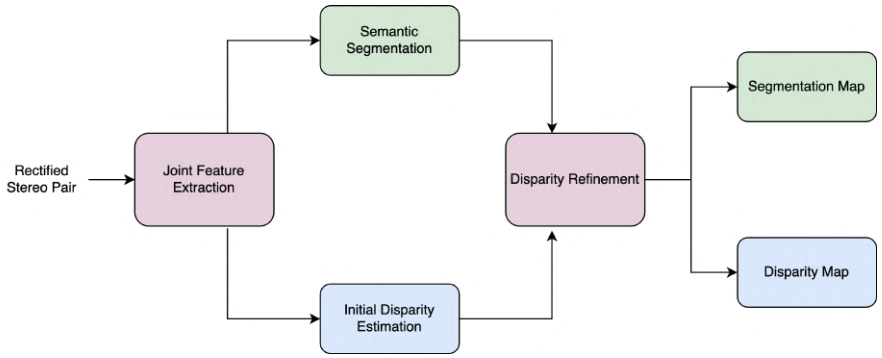


Figure 1.5: Generic features are extracted from the stereo pairs in the Joint Feature Extraction stage. The output of such stage is fed into two neural networks that take care for Disparity Estimation and Semantic Segmentation. The initial disparity obtained is then refined using semantic cues in Disparity Refinement stage.

Semantic depth estimation networks can be either:

- **Supervised**, in which datasets with stereo images and disparity ground truth are required, since the network loss is reduced by comparing the output of the network with the ground truth;
- **Unsupervised**, in which no ground truth is required, as they mainly rely on warping error.

The overall loss function of such networks is given by:

$$\mathcal{L} = \alpha\mathcal{L}_{\text{disp}} + \beta\mathcal{L}_{\text{seg}} + \gamma\mathcal{L}_{\text{ref}}$$

where $\mathcal{L}_{\text{disp}}$ is the initial disparity loss, $\mathcal{L}_{\text{seg}}$ is the segmentation loss and $\mathcal{L}_{\text{ref}}$ is the refined disparity loss [52]. Such loss functions depend on the approach employed. α, β and γ are empirically determined constraints.

1.1.3.2 End-to-end deep networks

End-to-end approaches paved the way for the most popular architectures nowadays, such as HSM-Net [54], LEAStereo [8], CFNet [38] and RAFT-Stereo [16]. These networks differ in architecture and thus in problems that address.

HSM-Net HSM-Net is based on a widely employed encoder-decoder architecture and addresses the issues of memory occupancy and speed limitation, related to the processing high-resolution images. Its approach allows to control the performance-vs-speed trade-off, addressing sensing needs for time-critical applications such as autonomous driving. The net leverages on its hierarchical architecture to retrieve on-demand reports of disparity by capping intermediate coarse results, allowing accurate prediction disparity for near-range structures with low latency. An idea of the general architecture and an example of on-demand report [54] is shown in Figure 1.6.

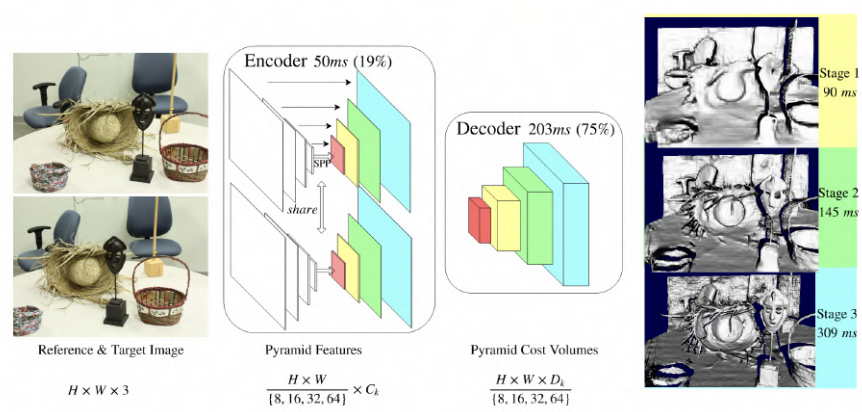


Figure 1.6: Given a rectified stereo pair of high-resolution images, multiscale descriptors are computed with a residual butterfly encoder-decoder neural network. Such descriptors are employed to construct 4D feature volumes at each scale by taking the difference of potentially matching features extracted from epipolar scanlines. The decoded output is used to predict 3D cost volumes that generates on-demand disparity estimates at a given scale and upsampled so that it can be combined with the next feature volume in the pyramid.

This approach allows to control the performance-vs-speed trade-off, addressing sensing needs for time-critical applications such as autonomous driving.

LEAStereo LEAStereo relies on Neural Architecture Search (NAS), a technique that enables the network to choose among a set of operations, such as convolution with different filter sizes or multi-layer perceptron with different numbers of hidden layers, to find an optimal architecture that is better suited for the task. Before LEAStereo, capabilities of NAS had not been exploited by low-level geometric vision tasks such as stereo matching, mainly due to the fact that state-of-the-art deep stereo matching networks are already sheer in size. Indeed, directly applying a NAS to such massive structures is com-

putationally prohibitive, based on most of the currently available computing resources.

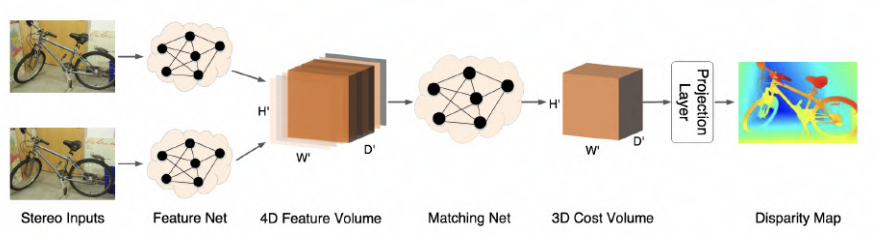


Figure 1.7: Stereo matching pipeline employing Neural Architecture Search.

LEAStereo applies NAS separately in each sub-module [8], as depicted in Figure 1.7, jointly optimizing the architecture.

CFNet CFNet aims to improve the robustness of the stereo matching network by employing a fused cost volume representation to mitigate the large domain differences present across a variety of datasets, that leads to poor generalization capabilities. By fusing multiple low-resolution dense cost volumes, an enlarged receptive field can be achieved in order to extract robust structural representations for initial disparity estimation. Furthermore, a cascade cost volume representation is devised to alleviate the unbalanced disparity distribution, that elicits distorted and noisy learned features. In particular, a variance-based uncertainty estimation which adaptively adjust the next stage disparity search space is employed, driving the network to progressively prune out the space of unlikely correspondences. The architecture [38] of such neural network is shown in Figure 1.8.

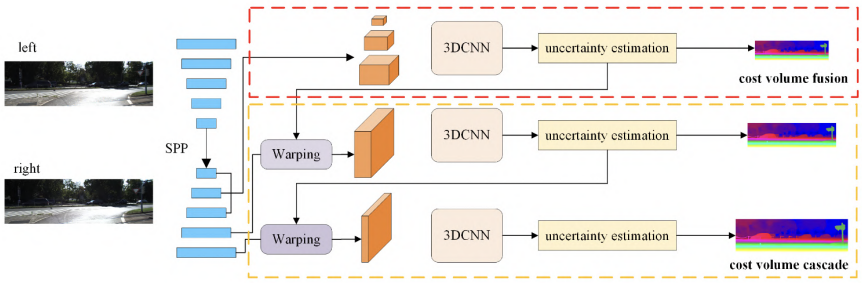


Figure 1.8: The network consists of three components: pyramid feature extraction, fused cost volume, cascade cost volume.

RAFT-Stereo RAFT-Stereo is based on a popular deep neural architecture for optical flow estimation, re-adapted for stereo matching. Optical flow and rectified stereo are closely related tasks. In optical flow, the aim is to predict a pixel-wise displacement field, such that for every pixel in the first frame, it is possible to estimate its correspondence in the second frame. In stereo matching, the task is the similar, except for the additional constraints that the x-displacement is always positive and the corresponding points lie on a horizontal line. Despite the similarities between stereo and flow, the architectures for the two tasks are vastly different. The architecture of RAFT-Stereo [16] is depicted in Figure 1.9.

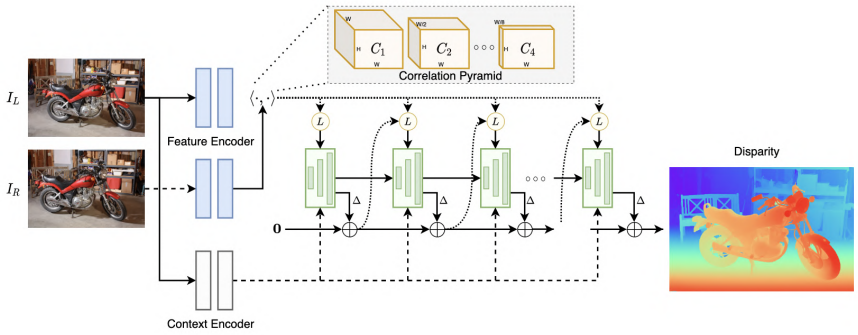


Figure 1.9: Correlation features are extracted from each image and are used to construct the correlation pyramid. Context image features and an initial hidden state are also extracted from the context encoder. The disparity field is initialized to zero everywhere. At each iteration, the GRUs use the current disparity estimate to sample from the correlation pyramid. The resulting correlation features, initial image features and current hidden states are employed by the GRUs to generate a new hidden state and an update to the disparity.

In stereo, the predominant approach has been the use of 3D convolutional neural networks, while optical flow is approached using iterative refinement. In particular, RAFT shows that iterative refinement can be performed entirely at high resolution, allowing also to control the accuracy-vs-efficiency trade-off with early stopping.

Although the aforementioned networks delivers impressive results in stereo matching, they still falters in presence of non-Lambertian surfaces. However, the various approaches used by these different networks can be combined to achieve a more robust model in terms of speed, precision and ability to generalize.

1.2 Novel View Synthesis

Novel View Synthesis (NVS) is another long-standing problem at the intersection between Computer Graphics and Computer Vision [24]. NVS techniques aim to generate novel views of a target scene, with an arbitrary camera pose, from one or more given source images and their camera poses.

1.2.1 Classical methods

Earlier works either exploit inter-image correspondences to estimate depth maps or use voxel grids to represent shapes. The former methods fuse depth maps into point clouds, relying heavily on the quality of matching and often producing errors which are hard to rectify during post-processing. The latter methods estimate occupancy and color for each voxel, though, they are usually limited by cubic memory requirements.

1.2.2 Explicit surface representations

Following works proposed explicit surface representations to estimate 3D mesh from images. First approaches assumed a fixed mesh topology, while several newer methods directly optimizes the surface mesh by means of differentiable marching tetrahedral layers [37]. Although such methods usually requires training with 3D supervision, approaches that leverage on 2D supervision — which also support inverse rendering — are being explored [24].

1.2.3 Implicit neural representations

Once again, deep learning allowed tremendous improvements in term of quality of the results, bringing renewed popularity to the field. Implicit neural representations leverage on differentiable rendering to reconstruct 3D geometry with appearance from image collections with different viewpoints.

1.2.3.1 Neural Radiance Fields

The core of this approach is represented by Neural Radiance Fields (NeRF). NeRF is a popular view synthesis technique that represents a scene as a continuous volumetric function, parameterized by multi-layer perceptrons

that provide the volume density and view-dependent emitted radiance at each location [22].

This model represents a static scene as a continuous 5D function that outputs the radiance emitted in each direction at each point in space, and a density at each point, which acts like a differential opacity controlling how much radiance is accumulated by a ray passing through the point [22]. The method optimizes a multi-layer perceptron to represent this mapping by regressing from a single 5D coordinate to a single volume density and view-dependent RGB color. Then, classical volume rendering techniques are employed to accumulate colors and densities into a 2D image. Such process is depicted in Figure 1.10.

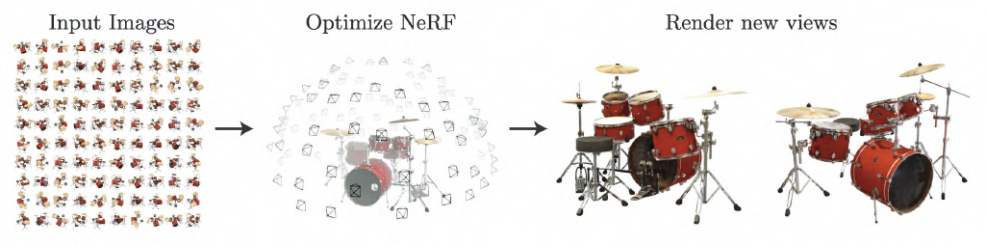


Figure 1.10: The main idea behind a NeRF-based model pipeline.

Thanks to the outstanding success of this technique a long series of iterations has been proposed.

Quality-related enhancements Although NeRF has demonstrated impressive results in view synthesis, the rendering model is flawed in a manner that may cause excessive blurring and aliasing [2]. The first research efforts were directed towards the enhancing of scene representation capabilities of NeRF.

Fully-connected deep neural networks are biased to learn low-frequency features, missing the opportunity to represents more detailed features. To overcome this spectral bias a simple mapping to the network input is able to mitigate this issue [22, 44, 51].

Another line of improvement, concerning aliasing, has been provided by mip-NeRF, which extends NeRF capabilities to represent the scene at a continuously-valued scale by efficiently rendering anti-aliased conical frustums instead of rays [2]. The benefit is even greater in situations where scene content is observed at different resolutions, for example in setups where the camera moves closer and farther from the scene.

Training and inference speed-up Beyond quality-related improvements, also strategies to speed-up training and inference have been explored.

The adoption of a multi-resolution hash encoding of the input permits the use of a smaller network without sacrificing quality, thus significantly reducing the number of floating point and memory access operations, enabling training of high-quality neural graphics primitives in a matter of seconds [23].

While training can be also sped up with a multi-GPU cluster, usually rendering must happen in real-time on smaller devices for interactive applications. KiloNeRF address such problem by using a large number of independent and small networks, letting each network represent only a fraction of the scene [27]. First a regular NeRF is trained as teacher model, then KiloNeRF is trained such that its outputs match those of the teacher model for any position and view direction and, at last, is fine-tuned on the original training images. This three-stage strategy allows the model to reach the same visual fidelity of original NeRF, while being able to synthesize novel views significantly faster.

Material editing capabilities Current research efforts point towards a common thread, namely relighting and material editing capabilities [17, 12, 24, 5, 57, 40, 51].

NeRF's view dependency can only handle simple reflections effects such as highlights, but cannot deal with complex reflections like those from glass and mirrors. NeRFReN splits the scene into transmitted and reflected components, modelling the two components with separate neural radiance fields, ensuring a better representation and enabling scene editing applications [12]. Since this decomposition is highly under-constrained, geometric priors and training strategies are exploited to ensure reasonable and physically sound decomposition results, enabling scene editing applications.

NeRV includes simulation of light transport, allowing rendering under arbitrary novel illumination conditions [40]. Instead of modeling a scene as a continuous 3D field of particles that absorb and emit light, NeRV represent a scene as a 3D field of oriented particles that absorb and reflect the light emitted by external light sources.

NeRF often fails to accurately capture and reproduce the appearance of glossy surfaces. This limitation is addressed by Ref-NeRF, which replaces NeRF's parameterization of view-dependent outgoing radiance with a representation of reflected radiance [51]. Such representation is structured using a collection of spatially-varying scene properties. With the aid of a regularizer

on normal vectors, the model significantly improves the realism and accuracy of specular reflections. Besides, model's internal representation of outgoing radiance is interpretable and useful for scene editing purposes.

NeRFactor distills the volumetric geometry of NeRF representation of the object into a surface representation and then jointly refine the geometry while solving for the spatially-varying reflectance and environment lighting [57]. The model recovers a 3D neural fields of surface normals, light visibility, albedo, and Bidirectional Reflectance Distribution Functions (BRDFs) without any supervision, by means of a re-rendering loss, smoothness priors, and a data-driven BRDF prior learned from real-world BRDF measurements [20]. Since it explicitly model light visibility, NeRFactor is able to separate shadows from albedo and synthesize realistic soft or hard shadows under arbitrary lighting conditions. It is possible to edit the albedo and the non-diffuse BRDF, then re-render the edited object under an arbitrary lighting condition from any viewpoint.

To sum up, NeRFactor factorizes images of an object under an unknown lighting condition into shape, reflectance, and illumination, hence supporting free-viewpoint relighting with shadows and material editing [57], as shown in Figure 1.11.

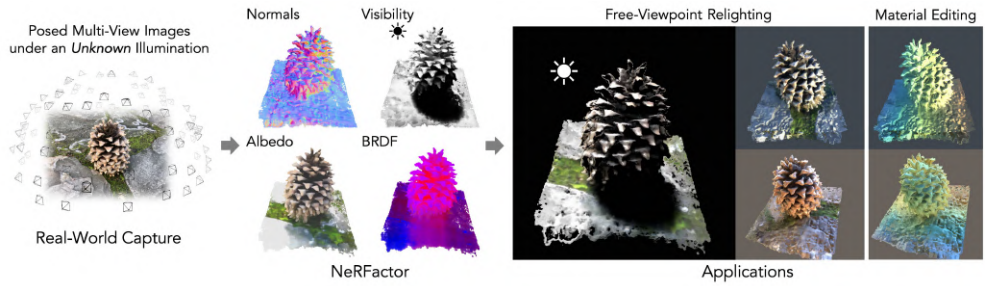


Figure 1.11: Example of a NeRFactor decomposition with subsequent editing.

1.3 Benchmarks

Objects with specular and transparent surfaces are almost absent or unlabeled in most stereo benchmarks.

In KITTI 2015 [21] cars have been replaced with CAD models providing supervision on some specular or transparent surfaces, however, such dataset is not enough to address the problem with deep learning.

ClearGrasp [30] includes several images featuring transparent elements. However, such dataset is (i) almost entirely synthetic, with the real-world segment consisting of very specific scenes; (ii) acquired employing a RGB-D mono setup. For the aforementioned reasons, such dataset is out of the scope.

ClearPose [7] is another large-scale dataset that includes transparent objects. Unlike the previous, it is exclusively based on real-world images. Nonetheless, it is acquired using a comparable mono setup and so falls out of the scope.

A recent work produced Booster [56], a novel dataset consisting in high-resolution stereo pairs featuring a large collection of labeled non-Lambertian objects. An example is depicted in Figure 1.12. The dataset is annotated in a semi-automatic fashion by employing a space-time framework [9] based on RAFT-Stereo [16] and subsequent manual filtering. However, Booster includes a rather small number of annotated images, and thus cannot be considered a large-scale dataset, needed for deep learning training.

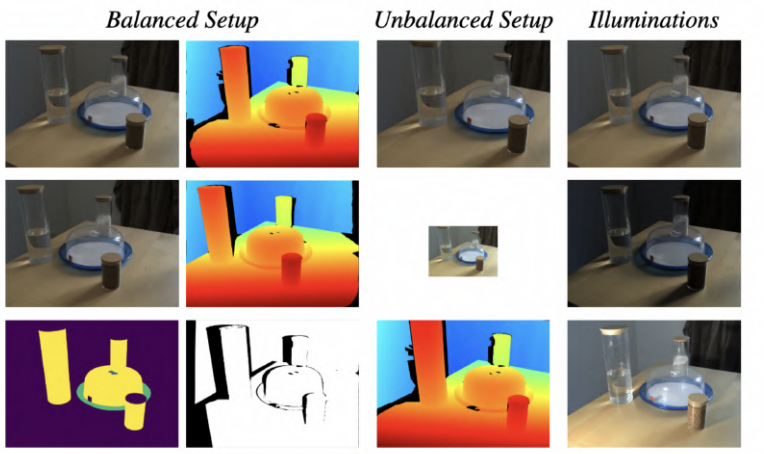


Figure 1.12: A scene from the Booster testing split.

Chapter 2

Frameworks for deep stereo matching

2.1 Multilevel Recurrent Field Transforms for Stereo Matching

As mentioned in Section 1.1.3.2, optical flow and stereo matching are strictly related tasks. Indeed, it is possible to translate the stereo matching problem formulation into the optical flow one and solve it with approaches adopted for the latter task.

RAFT-Stereo [16] is an example of end-to-end deep neural network for stereo matching based on RAFT [45], a network for optical flow.

Given a pair of rectified images (I_L, I_R), RAFT-Stereo aims to estimate a disparity field $\boldsymbol{d}$ representing the horizontal displacement for every pixel in I_L, with respect to I_R.

2.1.1 Architecture

Both networks are composed by (i) a feature extractor; (ii) a correlation pyramid; (iii) a GRU-based update operator. The architecture of RAFT-Stereo [16] is depicted in Figure 1.9.

2.1.1.1 Feature extractor

Two separate feature extractors are devised [16]:

- **Feature encoder**, applied to both images, maps each image to a dense feature map employed to construct the correlation volume. It consists in a sequence of residual blocks and downsampling layers with instance normalization;
- **Context encoder**, applied only on the left image, shares identical architecture but exploits batch normalization instead. Context features are employed to initialize the hidden state of the update operator and are also injected into the GRU during each iteration.

2.1.1.2 Correlation Pyramid

Similar to other neural stereo matching approaches, the dot product between feature vectors is utilized as a measure of visual similarity [16]. Given the feature maps $\boldsymbol{f}, \boldsymbol{g} \in \mathbb{R}^{H \times W \times D}$ extracted, by means of the aforementioned feature encoder, from I_L and I_R, the 3D correlation volume can be computed as follows:

$$\boldsymbol{C}_{ijk} = \sum_{h} \boldsymbol{f}_{ijh} \cdot \boldsymbol{g}_{ikh}, \quad \boldsymbol{C} \in \mathbb{R}^{H \times W \times W} \tag{2.1}$$

The computation of such 3D volume can be efficiently implemented using a single matrix multiplication, easily computed on GPUs.

Then, a four level pyramid of correlation volumes is constructed through repeated average pooling of the last dimension. The k-th level of the pyramid is built from the volume at level k employing 1D average pooling with a kernel and stride of size two, producing a new volume $\boldsymbol{C}^{k+1}$ with dimension $H \times W \times W/2^k$. With this method each level of the pyramid has an increased receptive field by only pooling the last dimension, maintaining high resolution information of the original image, allowing to recover very fine structures.

At last, a correlation lookup operator $L_{\mathbb{C}}$ is defined to index into the correlation pyramid. Given an estimate of disparity $\boldsymbol{d}$, a 1D grid with integer offsets around the current disparity estimate is built. Afterwards, such grid is used to index from each level in the correlation pyramid. Since these grid values are real numbers, a linear interpolation is employed when indexing each volume. Retrieved values are then concatenated into a single feature map. The whole process is depicted in Figure 2.1.

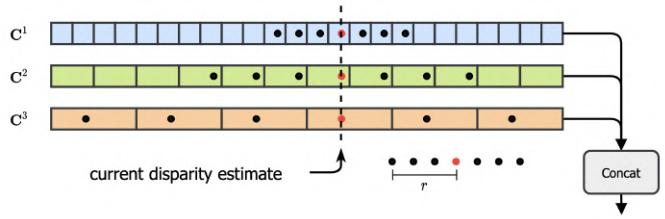


Figure 2.1: The current estimate of disparity is used to retrieve values from the each level of the correlation pyramid. Indexing from each level in the pyramid is performed by means of linear interpolation at the current disparity estimate and at integer offsets, whose size depends on the pyramid level.

2.1.1.3 Multi-Level Update Operator

The aim is to predict a series of disparity fields $\boldsymbol{d}_1, \ldots, \boldsymbol{d}_N$ from an initial disparity field $\boldsymbol{d}_0 = \mathbf{0}$ [16, 45].

At each iteration, the current estimate of disparity is used to index the correlation volume, producing a set of correlation features. These features are passed through a couple of convolutional layers, while similarly also the current disparity estimate is passed through a couple convolutional layers. The correlation volume $\boldsymbol{C}$ with its features $\boldsymbol{f}$, the current disparity $\boldsymbol{d}_{i-1}$, and context features $\boldsymbol{c}$ are then concatenated and injected into the GRU Θ, that updates its hidden state [56, 16, 45]:

$$\boldsymbol{d}_i = \Theta(\boldsymbol{f}, \boldsymbol{c}, \boldsymbol{d}_{i-1}, \boldsymbol{C}) \tag{2.2}$$

Finaly, the new hidden state is used to predict a new disparity update $\boldsymbol{d}_i$.

2.1.2 Supervision

The supervision is based on the ℓ_1 distance between the predicted and ground truth disparity over the full sequence of predictions, $\boldsymbol{d}_1, \ldots, \boldsymbol{d}_N$, with exponentially increasing weights, to give more importance to more recent, hence finer, predictions [16, 45]. Given the ground truth disparity $\boldsymbol{d}_{gt}$, the loss is defined as follows:

$$\mathcal{L} = \sum_{i=1}^{N} \gamma^{N-1} ||\boldsymbol{d}_{gt} - \boldsymbol{d}_i||_1$$

where γ is a hyperparameter, usually empirically fixed at $\gamma = 0.9$.

2.1.3 Main advantages

The main advantages of this architecture are:

(i) State-of-the-art cross-dataset generalization;

(ii) Efficient high-resolution processing;

(iii) Possibility to perform accurate real-time inference.

2.2 Space-time stereo framework

In Section 1.1.1 a primary classification of active and passive methods has been depicted. Active techniques, such as laser scanning and structured light, project illumination into the scene in order to construct easily identifiable features to mitigate the difficulty involved in determining correspondence, while passive stereo algorithms attempt to find matching image features between a pair of general images with no prior information.

However, there exists another popular taxonomy of algorithms for depth from triangulation, based on the domain in which corresponding features are located [9]:

- **Spatial domain**, in which correspondence is found by determining similarity of pixels in the image plane, such as in traditional laser scanning and passive stereo;
- **Temporal domain**, in which features with similar appearance over time are likely to correspond, such as in coded structured light and temporal laser scanning.

Most methods locate features entirely within either the spatial or temporal domains. Nonetheless, it is possible to locate features within both the space and time domains, by means of the general framework of space-time stereo, that grants greater flexibility, accuracy and robustness.

2.2.1 Spatial stereo

The space-time stereo framework can be seen as a generalization of traditional passive stereo methods, which operate exclusively within the spatial

domain. Such method considers two viewpoints in known positions, and attempts to find corresponding pixels in each of the two images. This search for correspondence can performed either by (i) searching for specific features in each of the images; (ii) matching of arbitrary spatial windows in the first image to corresponding regions along the epipolar line in the second image.

More specifically, spatial stereo minimizes a generic matching function [9]:

$$\Phi_s = \left|\left| I_L\big(\mathcal{N}_s(x_L)\big) - I_R\big(\mathcal{N}_s(x_R)\big) \right|\right|^2 \tag{2.3}$$

where I_L is the intensity in the left image, I_R is the intensity in the right, and $\mathcal{N}_s$ is a vector of pixels in a spatial neighbourhood close to x.

As already mentioned in Section 1.1.2.7, there is a trade-off in determining the size of the neighbourhood to use:

- If the window is too small there could be many pixels along the epipolar line that match the pixel in the left image equally well;
- If the window is too large more information is not guaranteed, since there could be texture less regions. Moreover, depth discontinuities may be encountered.

Due to this trade-off, traditional stereo methods may lack robustness and return low quality dense depth estimates.

2.2.2 Temporal stereo

In temporal stereo, a scene with static geometry viewed for multiple frames across time is considered. In this setup, once again, pixels from the left image are matched against pixels on the right image. However, rather than considering a neighbourhood in the spatial direction, it is possible to consider a neighbourhood in the temporal direction, hence considering the same windows in different time frames [9].

Analogously to the spatial case, temporal stereo minimizes a similar generic matching function:

$$\Phi_t = \left|\left| I_L\big(\mathcal{N}_t(x_L, t_0)\big) - I_R\big(\mathcal{N}_t(x_R, t_0)\big) \right|\right|^2 \tag{2.4}$$

where $\mathcal{N}_t$ is a vector of pixels in a temporal neighbourhood close to x, around a central time t_0. Such vector could potentially contain more disambiguating information than a spatial matching vector, since the previously mentioned trade-off is no longer involved.

The differences between these two approaches are illustrated in Figure 2.2.

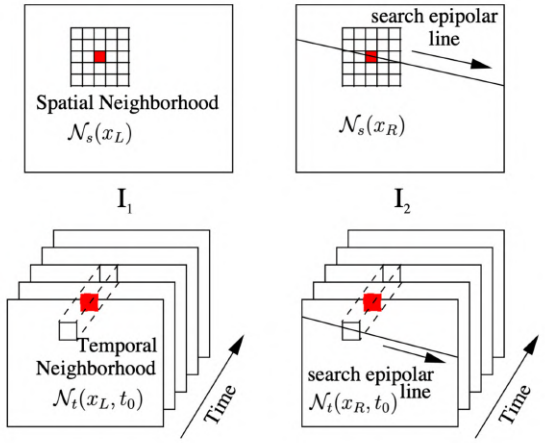


Figure 2.2: In spatial stereo the epipolar line is searched through similar spatial neighborhoods, while in temporal stereo the search is performed for similar temporal variation.

2.2.3 Space-time stereo

It is possible combine space and time frameworks by matching features along both spatial and temporal axes [9]. Considering a rectangular patch as neighbourhood $\mathcal{N}_{st}$, a window of size $N \times M \times T$ can be chosen, where N and M are the spatial sizes of the window and T is the dimension along the time axis, namely the number of frames considered.

Space-time stereo minimizes a generic matching function, based on Equation 2.3 and Equation 2.4:

$$\Phi_{st} = \left|\left| I_L\big(\mathcal{N}_{st}(x_L, t_0)\big) - I_R\big(\mathcal{N}_{st}(x_R, t_0)\big) \right|\right|^2 \tag{2.5}$$

There is no mathematical distinction between the spatial and temporal axes in Equation 2.5.

2.2.4 Spatial and temporal domain errors

Matching errors may arise in both spatial and temporal domains, and there is once again a trade-off in determining the size of the neighbourhood to use [9].

In spatial stereo matching, untextured regions create ambiguities, since increasing the size of the matching vector does not introduce new information to disambiguate potential matches. In temporal stereo matching, scenes

with constant illumination over time do not introduce any new information as well. Thus, spatial matching performs best on objects textured with high spatial frequency, while temporal matching performs best when the scene illumination has high temporal frequency.

In spatial stereo matching, depth discontinuities between objects create matching neighbourhoods with separate regions which cannot be matched. In temporal stereo matching, moving objects may cause the same sort of discontinuity.

The space-time stereo framework gives rise to the question of optimal spatial-temporal window size, which is scene and lighting dependent.

2.3 Booster approach

Booster employs a deep space-time stereo pipeline, based on RAFT-Stereo, to infer a dense and accurate disparity map. The approach is grounded by the fact that in presence of a distinctive colorful texture projected in the scene, the deep network can correctly infer a reliable disparity map.

2.3.1 Deep space-time stereo processing

Since T stereo pairs are available, it is possible to build an accumulated correlation volume $\boldsymbol{C}^*$, based on Equation 2.1, by averaging the correlation volumes computed from $\boldsymbol{f}^t$ and $\boldsymbol{g}^t$, extracted from a stereo pair t:

$$\boldsymbol{C}^*_{ijk} = \frac{1}{T} \sum_t \sum_h \boldsymbol{f}^t_{ijh} \cdot \boldsymbol{g}^t_{ikh}, \quad \boldsymbol{C}^* \in \mathbb{R}^{H \times W \times W}$$

This enriched volume can be exploited, analogously as done with the operator depicted in Equation 2.2, to estimate a set of disparity maps from any given stereo pair:

$$\boldsymbol{d}^t_i = \Theta(\boldsymbol{f}^t, \boldsymbol{c}^t, \boldsymbol{d}^t_{i-1}, \boldsymbol{C}^*) \tag{2.6}$$

Once the disparity maps $\boldsymbol{d}^t$ have been estimated, it is possible to compute their average to obtain an initial ground-truth disparity map $\boldsymbol{d}^*$, as well as an uncertainty guess $\boldsymbol{u}^*$ as their variance.

2.3.2 Super-resolution and sharpening

The quality of the disparity labels produced by the aforementioned process may be dampened by two main causes [56]: (i) the downsampled resolution; (ii) the presence of over-smoothed depth discontinuities, a common issue in disparity maps inferred by deep networks. To mitigate these issues a neural disparity refinement architecture is employed [1]. Various instances of such network are overfit on each scene, taking the disparity map as both input and ground-truth, allowing to preserve accurate disparity values at high-resolution, while sharpening depth boundaries.

Moreover, the sub-pixel prediction mechanism described is replaced by a Stereo Mixture Density (SMD) head [48], to avoid undesired artifacts observed in the former formulation [1]. Each neural disparity refinement network is optimized to infer a bimodal Laplacian distribution:

$$\boldsymbol{p}(d) = \frac{\pi}{2\boldsymbol{b}_1} e^{-\frac{\boldsymbol{d}^* - \mu_1}{\boldsymbol{b}_1}} + \frac{1-\pi}{2\boldsymbol{b}_2} e^{-\frac{\boldsymbol{d}^* - \mu_2}{\boldsymbol{b}_2}}$$

Once the network is trained, a sharpened disparity map $\boldsymbol{d}^*$ is obtained at full resolution by leveraging the continuous representation enabled by the refinement network, selecting the mode with highest density value [56].

2.3.3 Manual cleaning and filtering

Once a full-resolution disparity map has been obtained, any remaining artefact is manually cleaned [56]. The disparity map is projected into a 3D point cloud to better visualize structural errors in the geometry of the scene, while the variance map $\boldsymbol{u}^*$ is used as a guidance during this operation, to easily detect most of the artifacts. Points removed from the point cloud are filtered out from the disparity map as well. At last, a bilateral filter is employed to smooth objects surfaces and the final map $\boldsymbol{d}^*$ is obtained. The process depicted so far is shown in Figure 2.3.

2.3.4 Accuracy assessment

The accuracy of ground-truth annotations are measured by (i) manually selecting planar regions from the images; (ii) fitting a plane to the recovered disparities over each of them; (iii) measuring the residuals between the fitted plane equation and the actual disparities [33].

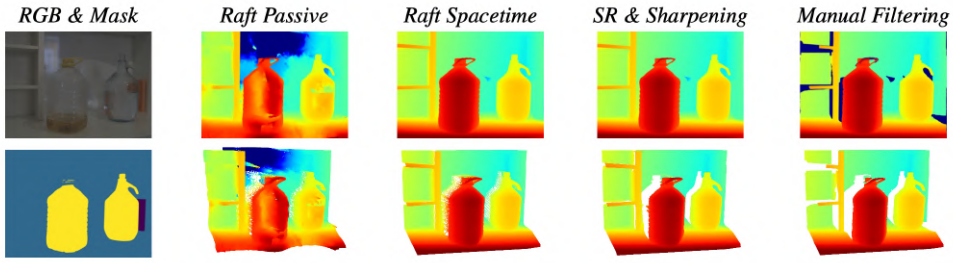


Figure 2.3: Data annotation pipeline.

2.3.5 Left-right consistency check

The processing pipeline is performed twice for each scene, producing two disparity maps, $\boldsymbol{d}_L^*$ and $\boldsymbol{d}_R^*$, for the left and right images respectively. Any pixel at coordinates (x, y) in $\boldsymbol{d}_L^*$ is filtered out in case the absolute difference with its match $\big(x - \boldsymbol{d}_L(x, y), y\big)$ in $\boldsymbol{d}_R^*$ is larger than a pixel threshold ξ:

$$\big|\boldsymbol{d}_L(x, y) - \boldsymbol{d}_R\big(x - \boldsymbol{d}_L(x, y), y\big)\big| > \xi$$

The same process is repeated on top of $\boldsymbol{d}_R$, removing any pixel at coordinate (x, y) after comparison with pixel $\big(x + \boldsymbol{d}_R(x, y), y\big)$ on the left disparity map.

Chapter 3

Novel View Synthesis with Neural Radiance Fields

Quick advancements in implicit neural representations are opening up new possibilities for Novel View Synthesis tasks. Such techniques can seamlessly reconstruct real objects, without requiring large amounts of data to learn from and without being limited to just a few points of view. It does this by learning a representation of a scene, using a sparse set of combined images from arbitrary viewpoints. Unlike traditional 3D representations, such as meshes or point clouds, this newer approach represents objects as a continuous function, allowing for more accurate reconstruction of shapes with complex geometries as well as higher colour reconstruction accuracy.

This research area is still in its embryonic stage, with new variants regularly emerging. Indeed, after the introduction of NeRF, more than fifty variants of this method have been published in the past year alone.

Most current neural implicit reconstruction methods create real-time photo-realistic renderings via ray marching. With this method, rays are emitted from the rendering camera and 3D points are sampled along these rays. Then, an implicit shape function, which represents the shape and appearance of the scene, evaluates density or distance to the surface at the sampled ray points. Next, a renderer marches along the ray points to find the first intersection between the scene's surface and the ray, in order to render image pixels. Finally, a loss functions between generated and ground-truth images is computed, along with other metrics.

3.1 Neural Radiance Fields

In Neural Radiance Fields (NeRF) a continuous scene is represented by means of 5D vector-valued function, whose input are [22]:

- A 3D location $\boldsymbol{x} = (x, y, z)$;
- A 2D viewing direction $\boldsymbol{d} = (\theta, \phi)$[1];

and whose output are;

- An emitted colour $\boldsymbol{c} = (r, g, b)$;
- A volume density σ.

The continuous 5D scene representation is approximated by means of an multi-layer perceptron (MLP)

$$F_\Theta : (\boldsymbol{x}, \boldsymbol{d}) \rightarrow (\boldsymbol{c}, \sigma)$$

whose weights Θ are optimized to map from each input 5D coordinate to its corresponding volume density and directional emitted color.

To encourage a multi-view consistency in the representation, the network is restricted to predict the volume density σ exclusively as function of the location $\boldsymbol{x}$, while the RGB colour $\boldsymbol{c}$ can be predicted as a function of both location and viewing direction [22].

An overview of the general pipeline of a Neural Radiance Field scene representation with its differentiable rendering procedure [22] is depicted in Figure 3.1.

3.1.1 Volume rendering

This 5D continuous function represents a scene as the volume density and directional emitted radiance at any point in space. The colour of any ray passing through the scene is then rendered using principles from classical volume rendering.

The volume density $\sigma(\boldsymbol{x})$ can be interpreted as the differential probability of a ray terminating at an infinitesimal particle at location $\boldsymbol{x}$ [22]. The expected

[1] Actually, for practical implementation this direction is usually expressed as a 3D unit vector.

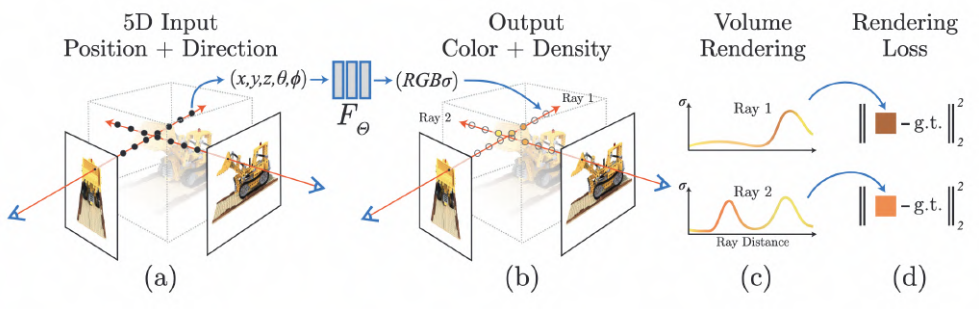


Figure 3.1: Images are synthesized by sampling 5D coordinates, both location and viewing direction, along camera rays (a). Feeding those locations into an MLP to produce a colour and volume density (b), and using volume rendering techniques is possible to composite these values into an image (c). Such rendering function is differentiable, so it possible to optimize this scene representation by minimizing the residual between synthesized and ground truth observed images (d).

colour $C(\boldsymbol{r})$ of camera ray $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$, with near and far bounds t_n and t_f such that $t \in [t_n, t_f]$, is given by:

$$C(\boldsymbol{r}) = \int_{t_n}^{t_f} T(t)\sigma\big(\boldsymbol{r}(t)\big)\boldsymbol{c}\big(\boldsymbol{r}(t), \boldsymbol{d}\big)dt \tag{3.1}$$

with

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\boldsymbol{r}(s))ds\right)$$

being the accumulated transmittance along the ray, from t_n to t. Such function represents the probability that the ray travels from t_n to t without hitting any other particle.

Rendering a view from this continuous representation requires estimating the integral in Equation 3.1, for a camera ray traced through each pixel of the desired virtual camera, using quadrature [22]. A stratified sampling approach is employed, where $[t_n, t_f]$ is partitioned into N evenly-spaced bins and then a single sample is drawn uniformly at random from within each bin. Although the set of samples to estimate the integral is discrete, stratified sampling enables a continuous scene representation since it results in the MLP being evaluated at continuous positions over the course of optimization.

Using quadrature, the continuous integral in Equation 3.1 is estimated as follows:

$$\hat{C}(\boldsymbol{r}) = \sum_{i=1}^{N} T_i\big(1 - \exp(-\sigma_i\delta_i)\big)\boldsymbol{c}_i \tag{3.2}$$

with

$$T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$$

and $\delta_i = t_{i+1} - t_i$ being the distance between adjacent samples. Such function is is trivially differentiable and reduces to traditional alpha compositing.

3.1.2 Optimization

A continuous scene representation and a classic rendering technique are not sufficient to achieve state-of-the-art quality. To enable high-resolution complex scenes representation two further components are needed: (i) a positional encoding of the input coordinates that assists the MLP in representing high-frequency functions; (ii) a hierarchical sampling procedure that permits an efficient sampling of such high-frequency representation.

3.1.2.1 Positional encoding

Neural networks are known as universal function approximators [15], though deep networks are biased towards learning lower frequency functions [26]. Indeed, having the network F_Θ directly operate on $xyz\theta\phi$ input coordinates results in renderings that perform poorly at representing high-frequency variation in color and geometry, making the networks incapable of representing complex scenes [22].

Mapping the inputs to a higher dimensional space using high frequency functions before passing them to the network enables better fitting of data that contains high frequency variation [26, 44]. Indeed, it is possible to reformulate F_Θ as a composition of two functions $F_\Theta = F'_\Theta \circ \gamma$, one learned and one not. In particular, γ is a mapping from $\mathbb{R}$ into a higher dimensional space $\mathbb{R}^{2L}$, and F'_Θ is still a regular MLP. In NeRF, the employed encoding is the so called positional encoding:

$$\gamma(p) = \left(\sin(2^0\pi p), \cos(2^0\pi p), \ldots, \sin(2^{2L-1}\pi p), \cos(2^{2L-1}\pi p)\right)$$

however, other encodings may be utilized [26, 44].

3.1.2.2 Hierarchical volume sampling

The rendering strategy of dense evaluation is inefficient since free space and occluded regions that do not contribute to the rendered image are still sam-

pled repeatedly. A hierarchical representation, that increase rendering efficiency by allocating samples proportionally to their expected effect on the final rendering, is employed.

Instead of using a single network, a coarse and a fine network are optimized. At first, a set of N_c locations is sampled, by means of the before mentioned stratified sampling, then the coarse network is evaluated at such locations, as depicted in Equation 3.2. Given the output of this coarse network, more informed sampling of points along each ray are produced, since these samples are biased towards the relevant parts of the volume.

To make it so, the alpha composited colour equation from the coarse network $\hat{C}_c(\boldsymbol{r})$, based on Equation 3.2, is reformulated as a weighted sum of all sampled colors c_i along the ray:

$$\hat{C}_c(\boldsymbol{r}) = \sum_{i=1}^{N_c} w_i c_i$$

with weights $w_i = T_i\big(1 - \exp(-\sigma_i \delta_i)\big)$. Normalizing these weights produces a piece-wise constant probability distribution function along the ray. A second set of N_f locations is sampled from this distribution using inverse transform sampling, then the fine network is evaluated at the union of the first and second set of samples and, in the end, the final rendered colour is computed for the ray $\hat{C}_f(r)$, using Equation 3.2, employing all $N_c + N_f$ samples.

Such procedure allocates more samples to regions expected to contain visible content.

3.1.3 Training and supervision

A separate neural continuous volume representation network is optimized for each scene. The goal is to overfit the network on the scene data. The training requires:

(i) A dataset of captured RGB images of the scene;

(ii) The corresponding camera poses;

(iii) The intrinsic parameters of the camera;

(iv) The scene bounds.

COLMAP [35, 36] is usually employed to obtain (ii), (iii) and (iv) starting from (i).

At each optimization iteration, a batch of camera rays from the set of all pixels in the dataset is randomly sampled. Then, the hierarchical sampling described in Section 3.1.2.2, is employed to query N_c samples from the coarse network and $N_c + N_f$ samples from the fine network. Next, the volume rendering procedure described in Section 3.1.1, is employed to render the color of each ray from both sets of samples. The loss function is simply the total squared error between the rendered and true pixel colours for both the coarse and fine renderings:

$$\mathcal{L} = \sum_{\boldsymbol{r}\in\mathcal{R}} \left[\left|\left| \hat{C}_c(\boldsymbol{r}) - C(\boldsymbol{r}) \right|\right|_2^2 + \left|\left| \hat{C}_f(\boldsymbol{r}) - C(\boldsymbol{r}) \right|\right|_2^2 \right] \tag{3.3}$$

where $\mathcal{R}$ is the set of rays in each batch, and $C(\boldsymbol{r})$, $\hat{C}_c(\boldsymbol{r})$, and $\hat{C}_f(\boldsymbol{r})$ are respectively the ground truth, coarse volume predicted, and fine volume predicted RGB colors for a ray $\boldsymbol{r}$. Even though the final rendering comes from the fine network with loss based on $\hat{C}_f(\boldsymbol{r})$, also the loss of $\hat{C}_c(\boldsymbol{r})$ is minimized since the weight distribution from the coarse network can be used to allocate samples into the fine network.

3.2 Neural Radiance Factorization of shape and reflectance under an unknown illumination

On of the major research direction in Novel View Synthesis with Neural Radiance Fields points towards granting these neural models editing capabilities [17, 12, 24, 5, 57, 40, 51]. To make it so is necessary to endow such models with some explainability mechanisms.

NeRFactor [57] is able to recover convincing relightable representations of an object captured under one unknown natural illumination condition, by (i) optimizing a Neural Radiance Field, to initialize the model's surface normals and light visibility; (ii) jointly optimizing these initial estimates along with the spatially-varying reflectance and the lighting condition, to best explain the observed images.

NeRF produces a high-quality geometry estimate for initialization, which helps to break the inherent ambiguities among shape, reflectance, and lighting, recovering a full 3D model for convincing view synthesis and relighting

using a re-rendering loss, simple spatial smoothness priors for each components, and a data-driven Bidirectional Reflectance Distribution Function (BRDF) prior.

Nevertheless, on its own NeRF suffers of two major issues that prevent it from being used for relighting [40, 57, 5]:

(i) NeRF models shape as a volumetric field, being computationally expensive to compute shading and visibility at each point along a camera ray for a full hemisphere of lighting;

(ii) The geometry estimated by NeRF contains extraneous high-frequency content that introduces high-frequency artifacts into the surface normals and light visibility, even though they are unnoticeable in view synthesis results.

NeRFactor addresses these issues by [57]:

(1) Using a hard surface approximation, performing shading calculations only at a single point along each ray, corresponding to the expected termination depth of the volume;

(2) Representing the surface normal and light visibility at any 3D location on the surface as continuous functions, parameterized by multi-layer perceptrons, encouraging such functions to be close to the values derived from the pre-trained NeRF and to be spatially smooth.

Neural Radiance Factorization [57] (NeRFactor) factors the observed images into estimated environment lighting as well as a 3D surface representation of the object with surface normals, light visibility, albedo, and spatially-varying BRDFs. Since NeRFactor models light visibility explicitly and efficiently, it is capable of removing shadows from albedo estimation and synthesizing realistic soft or hard shadows under arbitrary novel lighting conditions.

The structure of the network is depicted in Figure 3.2

3.2.1 Neural Radiance Factorization

NeRFactor takes as input a set multi-view images, with their camera poses, of an object lit by one unknown illumination condition. NeRFactor represents the shape and spatially-varying reflectance of the object as a set of 3D fields,

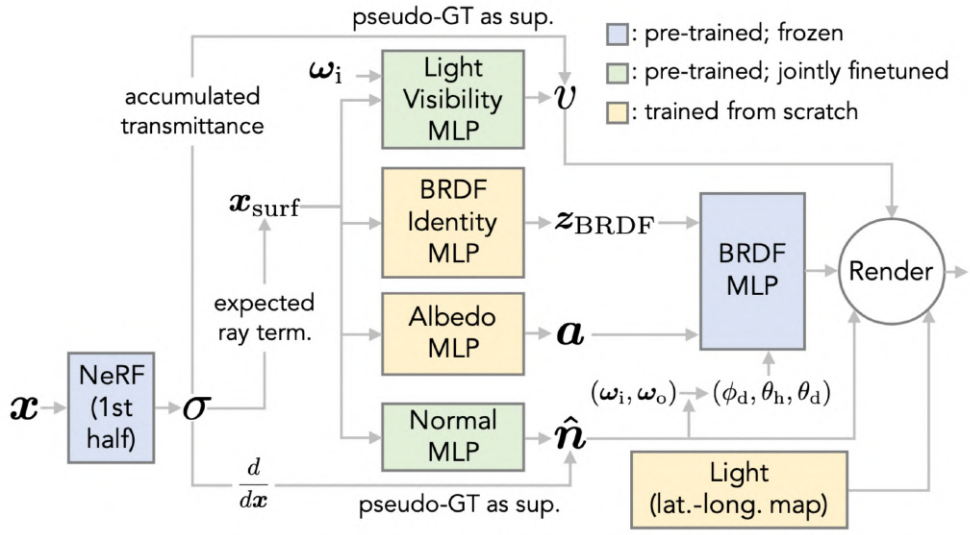


Figure 3.2: NeRFactor leverages NeRF's σ-volume as an initial estimate, to predict, for each surface location $\boldsymbol{x}_{\text{surf}}$: (i) surface normal $\hat{\boldsymbol{n}}$; (ii) light visibility v; (iii) albedo $\boldsymbol{a}$; (iv) BRDF latent code $\boldsymbol{z}_{\text{BRDF}}$; (v) lighting condition. $\boldsymbol{x}$ denotes 3D locations, $\boldsymbol{\omega}_i$ light direction, $\boldsymbol{\omega}_o$ viewing direction, while $(\phi_d, \theta_h, \theta_d)$ are the Rusinkiewicz coordinates. NeRFactor is an all-MLP architecture that models only surface points, unlike NeRF which models the entire volume.

each parameterized by multi-layer perceptrons, whose weights are optimized so as to explain the set of observed images.

After optimization, NeRFactor outputs, at each 3D location $\boldsymbol{x}$ on the object's surface: (i) the surface normal $\hat{\boldsymbol{n}}$; (ii) the light visibility in any direction $v(\boldsymbol{\omega}_i)$; (iii) the albedo $\boldsymbol{a}$; (iv) the reflectance $\boldsymbol{z}_{\text{BRDF}}$ [57]. These components explain the observed appearance, enabling applications such as free-viewpoint relighting with shadows and material editing.

3.2.1.1 Shape

NeRFactor leverages on NeRF's estimated geometry by distilling it into a continuous surface representation, used to initialize NeRFactor's geometry. An already optimized NeRF is employed to compute (i) the expected surface location along any camera ray; (ii) the surface normal at each point on the object's surface; (iii) the visibility of light arriving from any direction at each point on the object's surface [57].

Before the full optimization of NeRFactor, the visibility and normal MLPs

are pre-trained independently, to reproduce the visibility and normal values just from NeRF'σ-volume, without any smoothness regularization or re-rendering loss. This provides a reasonable initial estimate of the visibility maps, preventing the albedo MLP or the BRDF MLP from attempting to explain shadows as being modeled as painted on reflectance variation.

Surface points Given a camera and an already trained NeRF, the location at which a ray $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$ from that camera $\boldsymbol{o}$, along a direction $\boldsymbol{d}$ is expected to terminate according to NeRF's optimized volume density σ, is computed:

$$\boldsymbol{x}_{\text{surf}} = \boldsymbol{o} + \left(\int_0^\infty T(t)\sigma\big(\boldsymbol{r}(t)\big) t dt \right) \boldsymbol{d}$$

with

$$T(t) = \exp\left(- \int_0^t \sigma(\boldsymbol{r}(s)) ds \right)$$

being the probability that the ray travels distance t without being blocked.

Instead of maintaining NeRF's full volumetric representation, NeRFactor fix the geometry to lie on the surface distilled from the optimized NeRF, enabling much more efficient relighting during both training and inference, since it is possible to compute the outgoing radiance just at each camera ray's expected termination, instead of every point along each camera ray [57].

Surface normals Analytic surface normals $\hat{\boldsymbol{n}}_a(x)$ are computed as the negative normalized gradient of NeRF's σ-volume, with respect to x. However, normals derived from a trained NeRF tend to be noisy and therefore produce bumpy artifacts when employed for rendering. To mitigate this issue, these normals are re-parametrized with an MLP $\boldsymbol{f}_n : \boldsymbol{x}_{\text{surf}} \rightarrow \boldsymbol{n}$, which maps from any location $\boldsymbol{x}_{\text{surf}}$, on the surface to a denoised surface normal $\hat{\boldsymbol{n}}$ [57].

During the joint optimization of NeRFactor's weights, the output of this MLP $\boldsymbol{f}_n$ is encouraged to:

(i) Stay close to the normals produced from the pre-trained NeRF;

(ii) Vary smoothly in the 3D space;

(iii) Reproduce the observed appearance of the object.

Specifically, (i) and (ii) are enforced by employing the following loss function:

$$\mathcal{L}_n = \sum_{\boldsymbol{x}_{\text{surf}}} \left[\frac{\lambda_1}{3} \left|\left| \boldsymbol{f}_n(\boldsymbol{x}_{\text{surf}}) - \boldsymbol{n}_a(\boldsymbol{x}_{\text{surf}}) \right|\right|_2^2 + \frac{\lambda_2}{3} \left|\left| \boldsymbol{f}_n(\boldsymbol{x}_{\text{surf}}) - \boldsymbol{f}_n(\boldsymbol{x}_{\text{surf}} + \boldsymbol{\epsilon}) \right|\right|_1 \right]$$

where $\boldsymbol{\epsilon}$ is a random 3D displacement from $\boldsymbol{x}_{\text{surf}}$, sampled from a zero-mean Gaussian, and λ_1 and λ_2 are hyperparameters[2] [57]. Not restricting $\boldsymbol{x}$ on the expected surface increases the robustness of the MLP, by providing a safe margin where the output remains well-behaved even if the input is slightly displaced from the surface.

Light visibility The visibility v_a to each light source from any point is computed by marching through NeRF's σ-volume from the point to each light location. As to the estimated surface normals, the visibility estimates derived are too noisy to be used directly, since they would result in rendering artifacts. Once again, to mitigate this issue, the visibility is re-parametrized with an MLP $f_v : (\boldsymbol{x}_{\text{surf}}, \boldsymbol{\omega}_i) \rightarrow \boldsymbol{n}$, which maps from a surface location $\boldsymbol{x}_{\text{surf}}$ and a light direction $\boldsymbol{\omega}_i$, on the surface to the light visibility v [57].

The weights of f_v are optimized to encourage the recovered visibility field to:

(i) Be close to the visibility traced from the pre-trained NeRF;

(ii) Be spatially smooth;

(iii) Reproduce the observed appearance of the object.

Specifically, (i) and (ii) are enforced by employing the following loss function:

$$\mathcal{L}_v = \sum_{\boldsymbol{x}_{\text{surf}}} \sum_{\boldsymbol{\omega}_i} \left[\lambda_3 \big(f_v(\boldsymbol{x}_{\text{surf}}, \boldsymbol{\omega_i}) - v_a(\boldsymbol{x}_{\text{surf}}, \boldsymbol{\omega_i}) \big)^2 + \lambda_4 \big| f_v(\boldsymbol{x}_{\text{surf}}, \boldsymbol{\omega_i}) - f_v(\boldsymbol{x}_{\text{surf}} + \boldsymbol{\epsilon}, \boldsymbol{\omega_i}) \big| \right]$$

where $\boldsymbol{\epsilon}$ is a random 3D displacement from $\boldsymbol{x}_{\text{surf}}$, sampled from a zero-mean Gaussian, and λ_3 and λ_4 are hyperparameters [57]. Smoothness is encouraged across spatial locations given the same $\boldsymbol{\omega}_i$, to avoid the visibility at a certain location getting blurred over different light locations.

3.2.1.2 Reflectance

The BRDF model $\boldsymbol{R}$ consists of a diffuse Lambertian component fully determined by albedo $\boldsymbol{a}$ and a specular spatially-varying BRDF $\boldsymbol{f}_r$, defined for any location on the surface $\boldsymbol{x}_{\text{surf}}$ with incoming light direction $\boldsymbol{\omega}_i$ and outgoing direction $\boldsymbol{\omega}_o$, learned from real-world reflectance:

$$\boldsymbol{R}(\boldsymbol{x}_{\text{surf}}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \frac{1}{\pi} \boldsymbol{a}(\boldsymbol{x}_{\text{surf}}) + \boldsymbol{f}_r(\boldsymbol{x}_{\text{surf}}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$$

[2] The hyperparameters and the Gaussian's standard deviation are scene-dependent. However, in the original paper some fine-tuned parameters are listed for several types of scene.

NeRFactor starts with a learned reflectance function that is pre-trained to reproduce a wide range of empirically observed real-world BRDFs, while also learning a latent space for those real-world BRDFs. By doing so, the learn data-driven priors on real-world BRDFs encourage the optimization to recover plausible reflectance functions [57]. The use of such priors is crucial since the observed images are taken under an unknown illumination, making the problem highly ill-posed. Indeed, priors are necessary to disambiguate the most likely factorization of the scene from the set of all possible factorizations.

Albedo Also the albedo $\boldsymbol{a}$ is parameterized with an MLP $\boldsymbol{f}_a : \boldsymbol{x}_{\text{surf}} \rightarrow \boldsymbol{a}$, that maps any surface location $\boldsymbol{x}_{\text{surf}}$ to the albedo $\boldsymbol{a}$. Since there is no direct supervision, the model is only able to observe one illumination condition, relying on simple spatial smoothness priors and light visibility. In addition, the reconstruction loss of the observed views also drives the optimization of $\boldsymbol{f}_a$:

$$\mathcal{L}_a = \sum_{\boldsymbol{x}_{\text{surf}}} \frac{\lambda_5}{3} \left|\left| \boldsymbol{f}_a(\boldsymbol{x}_{\text{surf}}) - \boldsymbol{f}_a(\boldsymbol{x}_{\text{surf}} + \boldsymbol{\epsilon}) \right|\right|_1$$

where $\boldsymbol{\epsilon}$ is a random 3D displacement from $\boldsymbol{x}_{\text{surf}}$, sampled from a zero-mean Gaussian, and λ_5 is a hyperparameter. The output from $\boldsymbol{f}_a$ is employed as albedo in the Lambertian reflectance but not in the non-diffuse component, for which the specular highlight color is assumed to be white [57].

Learning priors from real-world BRDFs For the specular components of the BRDF, a latent space of real-world BRDFs is learned. Such latent space is paired with decoder that translates each latent code in the learned space $\boldsymbol{z}_{\text{BRDF}}$ to a full 4D BRDF, by means of a Generative Latent Optimization (GLO) approach [4]. The MLP $\boldsymbol{f}_r$ is pre-trained using the the MERL dataset [20], which isotropic materials. Thus, the incoming and outgoing directions for $\boldsymbol{f}_r$ are parametrized using Rusinkiewicz coordinates [28] $(\phi_d, \theta_h, \theta_d)$:

$$\boldsymbol{g} : (\hat{\boldsymbol{n}}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \rightarrow (\phi_d, \theta_h, \theta_d)$$

In the end, an MLP $\boldsymbol{f}_r^{'}$ that that maps from a concatenation of a latent code $\boldsymbol{z}_{\text{BRDF}}$, which represents a BRDF identity, and Rusinkiewicz coordinates $(\phi_d, \theta_h, \theta_d)$, to an achromatic reflectance $\boldsymbol{r}$, is trained:

$$\boldsymbol{f}_r^{'} : \left(\boldsymbol{z}_{\text{BRDF}}, (\phi_d, \theta_h, \theta_d)\right) \rightarrow \boldsymbol{r}$$

Both the weights of the MLP and the set of latent codes are optimized to reproduce a set of real-world BRDFs. Mean squared errors are computed

on the log of the High Dynamic Range (HDR) reflectance values to train $\boldsymbol{f}_r^{'}$ [57]. Since the colour component of the reflectance model is assumed to be handled by the albedo MLP, all colour information is discarded from the MERL dataset, by converting its RGB reflectance values into achromatic ones. Latent BRDF identity codes z_{BRDF} are parameterized as unconstrained 3D vectors and initialized with a zero-mean isotropic Gaussian.

After the pre-training, the weights of such BRDF MLP are frozen during the joint optimization. Only latent codes $\boldsymbol{z}_{\text{BRDF}}$ are predicted for each $\boldsymbol{x}_{\text{surf}}$ by training from scratch a BRDF identity MLP $\boldsymbol{f}_z : \boldsymbol{x}_{\text{surf}} \rightarrow \boldsymbol{z}_{\text{BRDF}}$. This can be thought of as predicting spatially-varying BRDFs for all the surface points in the plausible space of real-world BRDFs. This identity MLP is optimized to minimize the re-rendering loss and the same spatial smoothness prior as in albedo optimization:

$$\mathcal{L}_z = \sum_{\boldsymbol{x}_{\text{surf}}} \frac{\lambda_6}{\dim(\boldsymbol{z}_{\text{BRDF}})} \big|\big| \boldsymbol{f}_z(\boldsymbol{x}_{\text{surf}}) - \boldsymbol{f}_z(\boldsymbol{x}_{\text{surf}} + \boldsymbol{\epsilon}) \big|\big|_1$$

where $\boldsymbol{\epsilon}$ is a random 3D displacement from $\boldsymbol{x}_{\text{surf}}$, sampled from a zero-mean Gaussian, $\dim(\boldsymbol{z}_{\text{BRDF}})$ denotes the dimensionality of the latent code, and λ_6 is a hyperparameter [57].

The final BRDF model is the sum of the Lambertian component and the learned non-diffuse reflectance:

$$\boldsymbol{R}(\boldsymbol{x}_{\text{surf}}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \frac{1}{\pi} \boldsymbol{f}_a(\boldsymbol{x}_{\text{surf}}) + \boldsymbol{f}_r^{'}\Big(\boldsymbol{f}_z(\boldsymbol{x}_{\text{surf}}), \boldsymbol{g}\big(\boldsymbol{f}_n(\boldsymbol{x}_{\text{surf}}), \boldsymbol{\omega}_i, \boldsymbol{\omega}_o\big)\Big)$$

where the specular highlight color is assumed to be white.

3.2.1.3 Lighting

An HDR light probe image in the latitude-longitude format is employed as representation of lighting. Such representation permits to the model to represent detailed high-frequency lighting, hence to support hard cast shadows [57].

However, such representation is limited by a large number of parameters, which can vary independently of all other pixels. This issue can be mitigated by the use of the light visibility MLP, which allows a quick evaluation of surface point's visibility to all pixels of the light probe. To encourage smoother lighting, a ℓ_2 gradient penalty is enforced on the pixels of the light probe $\boldsymbol{L}$

along both the horizontal and vertical directions:

$$\mathcal{L}_i = \lambda_7 \Big|\Big| \begin{bmatrix} -1 & 1 \end{bmatrix} * \boldsymbol{L} \Big|\Big|_2^2 + \lambda_7 \Big|\Big| \begin{bmatrix} -1 \\ 1 \end{bmatrix} * \boldsymbol{L} \Big|\Big|_2^2$$

where $*$ denotes the convolution operator, and λ_7 is a hyperparameter. During the joint optimization, these probe pixels get updated directly by the final reconstruction loss and the gradient penalty [57].

3.2.1.4 Rendering

Given the surface normal, visibility for all light directions, albedo, BRDF at each point of the surface and the estimated lighting, the final physically-based non-learnable renderer renders an image that is compared against the observed image. The errors in this rendered image are backpropagated up to, excluding the σ-volume of the pre-trained NeRF, driving the joint estimation of surface normals, light visibility, albedo, BRDFs, and lighting. The rendering equation in this setup is:

$$\begin{aligned}
\boldsymbol{L}_o(\boldsymbol{x}_{\text{surf}}, \boldsymbol{\omega}_o) &= \int_\Omega \boldsymbol{R}(\boldsymbol{x}_{\text{surf}}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \cdot \boldsymbol{L}_i(\boldsymbol{x}_{\text{surf}}, \boldsymbol{\omega}_i) \cdot \big(\boldsymbol{\omega}_i \cdot \hat{\boldsymbol{n}} \boldsymbol{x}_{\text{surf}}\big) d\boldsymbol{\omega}_i \\
&= \sum_{\boldsymbol{\omega}_i} \boldsymbol{R}(\boldsymbol{x}_{\text{surf}}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \cdot \boldsymbol{L}_i(\boldsymbol{x}_{\text{surf}}, \boldsymbol{\omega}_i) \cdot \big(\boldsymbol{\omega}_i \cdot \hat{\boldsymbol{n}} \boldsymbol{x}_{\text{surf}}\big) \Delta\boldsymbol{\omega}_i \\
&= \sum_{\boldsymbol{\omega}_i} \left[\frac{1}{\pi} \boldsymbol{f}_a(\boldsymbol{x}_{\text{surf}}) + \boldsymbol{f}_r^{'} \Big(\boldsymbol{f}_z(\boldsymbol{x}_{\text{surf}}), \boldsymbol{g}\big(\boldsymbol{f}_n(\boldsymbol{x}_{\text{surf}}), \boldsymbol{\omega}_i, \boldsymbol{\omega}_o\big) \Big) \right] \cdot \\
&\qquad \boldsymbol{L}_i(\boldsymbol{x}_{\text{surf}}, \boldsymbol{\omega}_i) \cdot \big(\boldsymbol{\omega}_i \cdot \hat{\boldsymbol{n}} \boldsymbol{x}_{\text{surf}}\big) \Delta\boldsymbol{\omega}_i
\end{aligned}$$

where $\boldsymbol{L}_o(\boldsymbol{x}_{\text{surf}}, \boldsymbol{\omega}_o)$ is the outgoing radiance at $\boldsymbol{x}_{\text{surf}}$ as viewed from $\boldsymbol{\omega}_o$, $\boldsymbol{L}_i(\boldsymbol{x}_{\text{surf}}, \boldsymbol{\omega}_i)$ is the incoming radiance, masked by the visibility $\boldsymbol{f}_v(\boldsymbol{x}_{\text{surf}}, \boldsymbol{\omega}_i)$, arriving at $\boldsymbol{x}_{\text{surf}}$ along $\boldsymbol{\omega}_{\text{i}}$ directly from a light probe pixel, and $\Delta\boldsymbol{\omega}_i$ is the solid angle corresponding to the lighting sample at $\boldsymbol{\omega}_i$ [57].

The final reconstruction loss $\mathcal{L}_r$ is the mean squared error between the rendering and the observed image. The full loss function is the summation of all the previously defined losses:

$$\mathcal{L} = \mathcal{L}_r + \mathcal{L}_n + \mathcal{L}_v + \mathcal{L}_a + \mathcal{L}_z + \mathcal{L}_i$$

3.3 Instant Neural Graphics Primitives

Neural graphics primitives such as Neural Radiance Fields, parameterized by fully connected neural networks, can be computationally expensive to train and evaluate. As mentioned in Section 3.1.2.1, it is possible to devise several different encoding strategies in order to enhance the quality of the internal representation.

Instant Neural Graphics Primitives (NGP) [23] aims to reduce the computational cost of both training and inference with a versatile input encoding that allows the use of a smaller network, without sacrificing quality, thus significantly reducing the number of floating point and memory access operations. A small neural network is augmented by a multi-resolution hash table of trainable feature vectors whose values are optimized through gradient descent. Such structure allows the network to disambiguate hash collisions, making for a simple architecture that is trivial to parallelize on modern GPUs[3].

3.3.1 Multi-resolution Hash Encoding

Given a fully connected neural network $F_\Theta(y)$, the aim is to find an encoding of its inputs $y = \text{enc}_\theta(x)$ that improves the approximation quality of the representation and training speed of the network across a wide range of applications, without incurring a notable performance overhead. Such configuration not only has trainable weight parameters Θ, but also trainable encoding parameters θ. These parameters are arranged into L levels, each containing up to T feature vectors with dimension F [23].

Each level is independent and conceptually stores feature vectors at the vertices of a grid, which resolution N_l is chosen to be a geometric progression between the coarsest and finest resolutions $[N_{\min}, N_{\max}]$:

$$N_l \coloneqq \lfloor N_{\min} \cdot b^l \rfloor$$

with

$$b \coloneqq \exp\left(\frac{\ln N_{\max} - \ln N_{\min}}{L-1}\right)$$

and $N_{\max}$ being chosen to match the finest detail in the training data [23]. Due to the large number of levels L, the growth factor b is usually small.

[3]Source code available at https://github.com/nvlabs/instant-ngp.

Considering a single level l, each input coordinate $x \in \mathbb{R}^d$ is scaled by that level's grid resolution before rounding up and down:

$$\lceil x_l \rceil \coloneqq \lceil x \cdot N_l \rceil, \quad \lfloor x_l \rfloor \coloneqq \lfloor x \cdot N_l \rfloor$$

$\lceil x_l \rceil$ and $\lfloor x_l \rfloor$ span a voxel with 2^d integer vertices in $\mathbb{Z}^d$. Each corner of such voxel is then mapped to an entry in the level's respective feature vector array, which size is fixed to be at most T.

For coarse levels, in which a dense grid requires fewer than T parameters, this mapping is 1:1. For finer levels, an hash function $h : \mathbb{Z}^d \to \mathbb{Z}_T$ is employed to index into the array, treating it as an hash table with no explicit collision handling. A gradient-based optimization is devised to store appropriate sparse detail in the array, while the subsequent neural network $F_\Theta(y)$ takes care of collision resolution [23]. The number of trainable encoding parameters θ is therefore $\mathcal{O}(T)$ and bounded by $T \cdot L \cdot F$.

A spatial hash function is employed [46, 23]:

$$h(x) = \left(\bigoplus_{i=1}^{d} x_i \pi_i \right) \mod T \tag{3.4}$$

where $\oplus$ denotes the bit-wise XOR operation and π_i are unique, large prime numbers.

In Equation 3.4 the product behaves as a per-dimension linear congruential permutation (LCG), and the subsequent XOR decorrelates the effect of the dimensions on the hashed value. Notably, to achieve pseudo-independence it suffices to permute only $d - 1$ dimensions.

In the end, the feature vectors at each corner are d-linearly interpolated, according to the relative position of x within its hypercube, with an interpolation weight $w_l \coloneqq x_l - \lfloor x_l \rfloor$. The interpolated feature vectors of each level, as well as auxiliary inputs $\xi \in \mathbb{R}^E$, such as the encoded viewing direction and textures in neural radiance caching, are concatenated to produce $y \in \mathbb{R}^{LF+E}$, which is the encoded input $\text{enc}_\theta(x)$ to the MLP $F_\Theta(y)$.

The aforementioned process takes place independently for each of the L levels, and the general pipeline is depicted in Figure 3.3.

3.3.1.1 Performance-vs-quality trade-off

Choosing an hash table of size T provides a trade-off between performance, memory and quality. Indeed, higher values of T result in higher quality

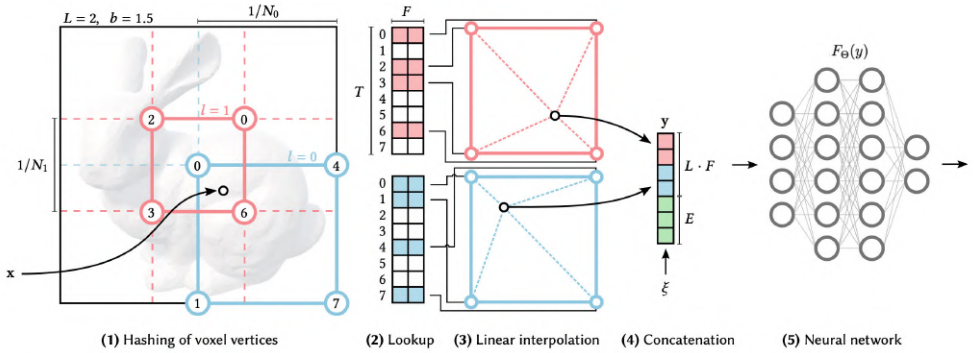


Figure 3.3: For a given input coordinate x, the surrounding voxels at L resolution levels are identified, and indices are assigned to their corners by hashing their integer coordinates (1). For all resulting corner indices, the corresponding F-dimensional feature vectors is looked up from the hash tables θ_l (2) and then is linearly interpolated according to the relative position of x within the respective l-th voxel (3). The result of each level, as well as auxiliary inputs $\xi \in \mathbb{R}^E$ are then concatenated, producing the encoded MLP input$y \in \mathbb{R}^{LF+E}$ (4), which is evaluated last (5). To train the encoding, loss gradients are backpropagated through the MLP (5), the concatenation (4), the linear interpolation (3), and then accumulated in the looked-up feature vectors.

and lower performance. The memory footprint is linear in T , while quality and performance tend to scale sub-linearly. The number of levels L and the number of feature dimensions F also trade-off quality and performance, for an approximately constant number of trainable encoding parameters θ.

3.3.1.2 Implicit hash collision resolution

Even though it may appear counter-intuitive, this encoding is able to reconstruct scenes faithfully in the presence of hash collisions, since different resolution levels have different strengths that complement each other.

Coarser levels, and thus the encoding, are injective, hence they do not suffer from collisions at all. However, they can only represent a low-resolution version of the scene, since they begets features which are linearly interpolated from a widely spaced grid of points. On the other hand, finer levels can capture small features due to their grid resolution, but they suffer from many collisions due to disparate points which hash to the same table entry[4].

[4]Nearby inputs with equal integer coordinates $\lfloor x_l \rfloor$ are not considered a collision. A collision occurs when different integer coordinates hash to the same index.

Such collisions are pseudo-randomly scattered across space and statistically unlikely to occur simultaneously at every level, for a given pair of points [23].

When training samples collide in this manner, their gradients average, but the importance to the final reconstruction of such samples is rarely equal. For instance, a point on a visible surface of a radiance field will contribute strongly to the reconstructed image [23]. Indeed, such point will have an high visibility and an high density, that multiplicatively affects the magnitude of gradients, causing large changes to its table entries. At the same time, a point in empty space that happens to refer to the same entry will have a much smaller weight. Thus, gradients of the more important samples dominate the collision average and the aliased table entry will naturally be optimized in such a way that it reflects the needs of the higher-weighted point.

The multi-resolution aspect of the hash encoding covers the full range from a coarse resolution $N_{\min}$, which is guaranteed to be collision-free, to the finest resolution $N_{\max}$ that the task requires. Doing so, it guarantees that all scales at which meaningful learning could take place are included, regardless of sparsity. The adopted geometric scaling allows covering these scales with only $\mathcal{O}\big(\frac{N_{\max}}{N_{\min}}\big)$ levels, permitting to pick a conservatively large value for $N_{\max}$.

3.3.1.3 Online adaptativity

If the distribution of inputs x changes over time during training, for instance if they become more concentrated in a small region, then finer grid levels will experience fewer collisions and thus a more accurate function can be learned. Indeed, the multi-resolution hash encoding automatically adapts to the training data distribution, inheriting the benefits of tree-based encodings [43, 23]. without task-specific data structure maintenance that may cause discrete leaps during training.

3.3.1.4 Interpolation

Interpolating the queried hash table entries ensures that the encoding $\mathrm{enc}_\theta(x)$ and, by the chain rule, its composition with the neural network $F_\Theta\big(\mathrm{enc}_\theta(x)\big)$ are continuous. Without this d-linear interpolation, grid-aligned discontinuities would be present in the network output, resulting in undesirable blocky appearances [23].

Chapter 4

Experiments on data augmentation

The main objectives of the thesis work are:

(i) Improving the quality of the underlying depth maps associated with NeRF-based models;

(ii) Exploiting the views synthesis capabilities of these models to generate novel multi-view datasets, containing challenging objects, associated with satisfactory depth maps;

(iii) Employing the aforementioned datasets as supervised data to augment the Booster dataset and, with the ultimate goal of fine-tuning RAFT-Stereo.

The first objective can be either realized by (i) modifying the underlying model, for instance by imposing constraints that emphasize the depth reconstruction; (ii) manipulating the physical scene, augmenting it with information that aids the reconstruction, for instance by using projectors to texturize the scene.

The experiments are mainly performed with Instant NGP thanks to its fastness in training and rendering.

4.1 Camera setup

To acquire images, the same custom stereo rig of the Booster method is employed.

The rig is made up of two high resolution cameras featuring a Sony IMX253LQR-C 12.4 Mpx sensor and a lower resolution camera equipped with a Sony IMX174LQJ-C 2.3 Mpx sensor mounted between the former two, as depicted in Figure 4.1.

Figure 4.1: Trinocular stereo rig on a tripod.

From left to right, (L, C, R) denote the three cameras, with L providing the reference image for both the balanced (L, R) and unbalanced (L, C) stereo pairs, and the baselines of these two setups being approximately 8 and 4 centimeters, respectively. However, in this application, the central low resolution camera is disregarded.

4.1.1 Camera calibration

Before the scene acquisitions, the rig has to be calibrated.

Individual cameras calibration At first each camera is calibrated separately using the pinhole camera model. For this purpose, N images, containing a known chessboard pattern, are acquired using the rig.

The distortion-free projective transformation performed by a pinhole camera model is given by:

$$p = A[\mathcal{R}|\mathcal{T}]P_w$$

where P_w is a 3D point expressed according to the world reference frame, p is a 2D pixel in the image plane, A is the intrinsic parameters matrix and $\mathcal{R}$, $\mathcal{T}$ are the rotation and translation from the world reference frame to the camera reference frame, respectively [56]. Following the OpenCV convention, which models lens distortion by means of a vector of parameters dist $= (k_1, k_2, k_3, p_1, p_2)$, k_1, k_2, k_3 denotes the radial distortion parameters and p_1, p_2 the tangential distortion parameters.

Given a chessboard, it is possible to find in the images a set of key-points, such as the inner corners of the chessboard, for which the exact 3D position in the world reference frame is known and, accordingly, build a set of 2D-3D correspondences. The 2D coordinates of the corners p_L, p_R in the L,R cameras are estimated by using a standard corner detection algorithm. By calibrating independently each camera of the rig, their intrinsic matrices A_L, A_R and the lens distortion parameters dist_L, dist_R are estimated [56].

Then images are undistort to perform a calibration of the stereo rigs, estimating also the rotation matrix $\mathcal{R}_{LR}$ and the translation vector $\mathcal{T}_{LR}$.

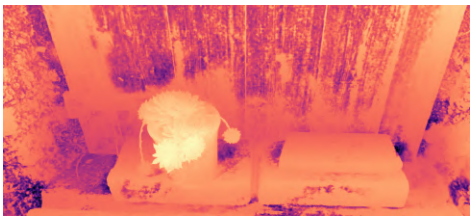
Stereo calibration After the individual camera calibrations, it is possible to estimate the rectification transformations to be applied to both images of the stereo rig to produce rectified stereo pairs. Since the $L-R$ stereo system is balanced, the problem can be addressed as a standard rectification since the resolution is the same for both images [56]. The OpenCV implementation is employed to estimate the new intrinsic matrices A_L^{LR}, A_R^{LR} and the rotations $\mathcal{R}_L^{LR}$, $\mathcal{R}_R^{LR}$ of L, R to map the initial image plane into the rectified image plane. This information is useful to obtain the L_{LR}, R_{LR} rectified stereo pair.

4.2 Depth supervision

On its own, NeRF-based models are unable to provide a good quality depth map for the whole scene. Indeed, such models tend to provide better results on areas of interest of the image, namely the objects in the scene, and worse results on the rest of the image, namely the background. An example is shown in Figure 4.2. This is probably due to the hierarchical representation embedded in these models, devised in Section 3.1.2.2, which allocates samples proportionally to their expected effect on the final rendering, for efficiency's sake.

Nonetheless, it is possible to add a further term in the loss function based

(a) Ground-truth RGB view.



(b) Generated disparity view without depth supervision.

Figure 4.2: Comparison between ground-truth RGB view and generated disparity view without depth supervision.

on Equation 3.3, presented in Section 3.1.3, that also takes into account a provided ground-truth, that can be either sparse or dense:

$$\mathcal{L} = \sum_{\boldsymbol{r}\in\mathcal{R}} \Big[\big|\big|\hat{C}_c(\boldsymbol{r}) - C(\boldsymbol{r})\big|\big|_2^2 + \big|\big|\hat{C}_f(\boldsymbol{r}) - C(\boldsymbol{r})\big|\big|_2^2 + \big|\big|\hat{\mathcal{D}}(\boldsymbol{r}) - \mathcal{D}(\boldsymbol{r})\big|\big|_1^2 \Big]$$

where $\mathcal{R}$ is the set of rays in each batch, while $\mathcal{D}(\boldsymbol{r})$ and $\hat{\mathcal{D}}(\boldsymbol{r})$ are respectively the ground-truth and predicted depth maps for a ray $\boldsymbol{r}$.

Since these scenes are manually acquired during the experiments, ground-truth depth maps are not available. However, it is possible to feed the stereo pairs to depth estimation models, such as RAFT-Stereo or SGM, in order to obtain a depth prior to employ as ground-truth for the supervision.

The process is devised as follows:

(i) Acquire m stereo pairs from different points of view;

(ii) Infer m left depth priors for each stereo pair, by employing a pre-trained model such as RAFT-Stereo;

(iii) Run COLMAP on the m left images acquired, to estimate the camera poses;

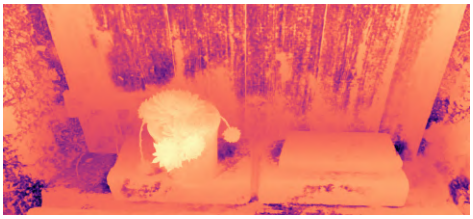
(iv) Train a model F_Θ taking as input the m left images previously acquired and the m left depth priors previously estimated;

(v) Render the depth maps for the points of view of interest, from the model F_Θ.

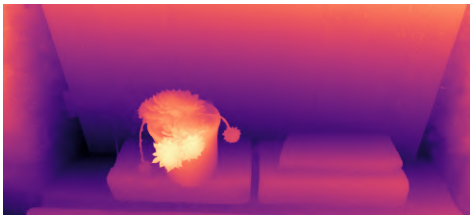
Note that the aforementioned process could be also performed by considering as input for the model both left and right images. However, due to hardware

limitations, for these experiments only the left images have been considered for the NeRF reconstruction.

Such method provides evident results for simple scenes that does not exhibits non-Lambertian effects, as depicted in Figure 4.3.



(a) Generated disparity view without depth supervision.



(b) Generated disparity with depth supervision.

Figure 4.3: Comparison between generated disparity view without supervision and with depth supervision.

4.3 Vanishing scanning spray

The same experiment can be replicated for a scene with objects which displays non-Lambertian behaviours. Nevertheless, the procedure is slightly different, since such objects must be covered in order to obtain good depth priors from the aforementioned depth estimation models:

(i) Acquire m stereo pairs from different points of view;

(ii) Run COLMAP on the m left images acquired, to estimate the camera poses;

(iii) Train a model F'_Θ taking as input the m left images previously acquired;

(iv) Paint the non-Lambertian objects with a vanishing scanning spray;

(v) Acquire n stereo pairs from different points of view;

(vi) Infer n left depth priors for each stereo pair of the painted scene, by employing a pre-trained model such as RAFT-Stereo;

(vii) Run COLMAP on the $m + n$ left images acquired, to estimate the camera poses;

(viii) Train a model F''_Θ taking as input the n left images previously acquired and the n left depth priors previously estimated;

(ix) Render the depth maps for the points of view of interest, namely the ones associated with the non-painted version of the scene, estimated in step (ii), from the model F''_Θ.

A result of such experiment is presented in Figure 4.4. Once again, it is self-evident that the disparity maps obtained without supervision are much worse with respect to the supervised ones. Moreover, there is the possibility to generate views of the unpainted version of the scene that match the painted version of the scene, thus the produced disparity maps.

4.4 Scene texturization

The idea is to speed up the process devised by the Booster approach, in Section 2.3, to obtain high-quality supervised stereo data. While the Booster approach infers high-quality depth maps directly from stereo pairs, the following approach aims to obtain depth maps from a NeRF-based reconstruction of multi-view stereo pairs.

Based on the previously performed preliminary experiments, a methodology has been devised. Given a scene containing objects which exhibit non-Lambertian behaviours, a calibrated stereo camera and k projectors, the general pipeline is the following:

(i) Acquire m passive stereo pairs from different points of view;

(ii) Run COLMAP on the m left images acquired, to estimate the camera poses;

(iii) Train a model F'_Θ taking as input the m left images previously acquired;

(iv) Paint the non-Lambertian objects with a vanishing scanning spray, to allow textures to be projected correctly;

(v) Acquire $2n$ stereo pairs from different point of views, acquiring a passive pair and an active pair from each point of view, employing all projectors at once;

(vi) Infer $2n$ depth priors for each stereo pair of the painted and texturized scene, by employing a pre-trained model such as RAFT-Stereo;

(a) Ground-truth RGB view with non-Lambertian objects painted with the vanishing scanning spray.

(b) Generated disparity view without depth supervision.

(c) Generated RGB view with unpainted non-Lambertian objects.

(d) Generated disparity view with depth supervision.

Figure 4.4: Results of the second experiment.

(vii) Run COLMAP on the $2(m+n)$ left images acquired, to estimate the camera poses;

(viii) Train a model F''_Θ taking as input the $2n$ images previously acquired and the $2n$ depth priors previously estimated;

(ix) Render the depth maps for the points of view of interest, namely the ones associated with the non-painted version of the scene, estimated in step (ii), from the model F_Θ.

The aforementioned process is performed by considering as input for the models both left and right images of the stereo pairs, in contrast to the previous experiment, since few images with texture have been acquired. Such methodology tends to provide better results.

In Figure 4.5, a stereo pair of the scene with not painted non-Lambertian objects and their respective depth maps, inferred with RAFT-Stereo, are shown. It is noticeable that most of non-Lambertian objects are inferred incorrectly.

In Figure 4.6, a stereo pair of the scene with painted non-Lambertian objects and their respective disparity maps, generated by training a NeRF model, without depth supervision, are presented. It is evident that, once again, most of non-Lambertian objects are inferred incorrectly. Moreover, the disparity maps generated by NeRF are much more noisy with respect to the ones inferred by RAFT-Stereo, shown before in Figure 4.5.

In Figure 4.7, a stereo pair of the scene with painted non-Lambertian objects and the respective stereo pair of the scene with painted and texturized non-Lambertian objects, are shown. The texturized version of the scene is then employed to generate, with RAFT-Stereo, a better depth prior for the depth supervision of the NeRF model.

In Figure 4.8, the pairs of the disparity maps, generated by training a NeRF model, both without and with depth supervision, are presented. This time is clear that the non-Lambertian objects are inferred correctly and without any major artifact.

In the end, it is also possible to generate the RGB views with unpainted non-Lambertian objects, as shown in Figure 4.9.

(a) Ground-truth RGB left view with not painted non-Lambertian objects.

(b) Ground-truth RGB right view with not painted non-Lambertian objects.

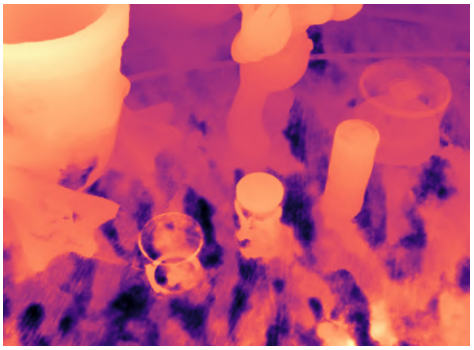
(c) RAFT-Stereo depth left view of not painted non-Lambertian objects.

(d) RAFT-Stereo depth right view of not painted non-Lambertian objects.

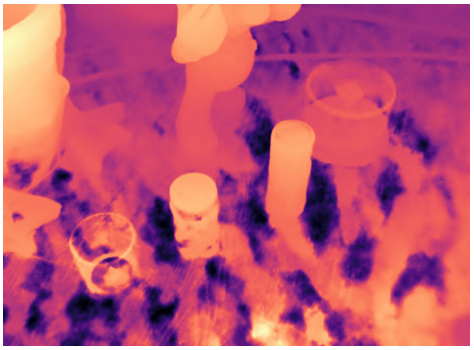
Figure 4.5: A stereo pair of the scene with not painted non-Lambertian objects and their respective RAFT-Stereo disparity maps.

(a) Ground-truth RGB left view with painted non-Lambertian objects.

(b) Ground-truth RGB right view with painted non-Lambertian objects.



(c) NeRF disparity left view of painted non-Lambertian objects, without depth supervision.



(d) NeRF disparity right view of painted non-Lambertian objects, without depth supervision.

Figure 4.6: A stereo pair of the scene with painted non-Lambertian objects and their respective NeRF disparity maps, without supervision.

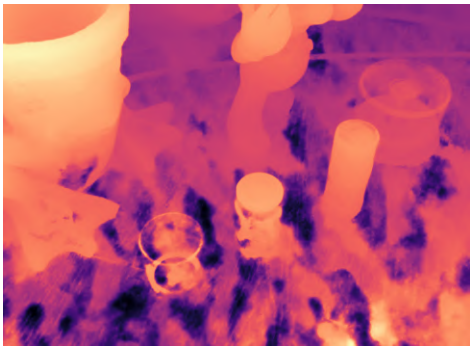
(a) Ground-truth RGB left view with painted non-Lambertian objects.

(b) Ground-truth RGB right view with painted non-Lambertian objects.

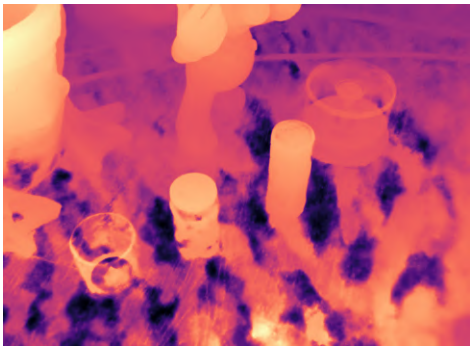
(c) Ground-truth RGB left view with painted and texturized non-Lambertian objects.

(d) Ground-truth RGB right view with painted and texturized non-Lambertian objects.

Figure 4.7: A stereo pair of the scene with painted non-Lambertian objects and the respective scene with texture projection.



(a) NeRF disparity left view of painted non-Lambertian objects, without depth supervision.



(b) NeRF disparity right view of painted non-Lambertian objects, without depth supervision.

(c) NeRF disparity left view of painted non-Lambertian objects, with depth supervision.

(d) NeRF disparity right view of painted non-Lambertian objects, with depth supervision.

Figure 4.8: Comparison between disparity maps generated by NeRF, without and with supervision, of a scene with painted non-Lambertian.

(a) Generated RGB left view with unpainted non-Lambertian objects.

(b) Generated RGB right view with unpainted non-Lambertian objects.

(c) NeRF disparity left view of painted non-Lambertian objects, with depth supervision.

(d) NeRF disparity right view of painted non-Lambertian objects, with depth supervision.

Figure 4.9: Generated supervised sample.

4.5 Results

Since the ground-truths of the acquired scenes are not available, the comparison between the disparity maps generated by RAFT-Stereo and the ones generated by NeRF has to be analyzed, in order to understand whether NeRF's depth estimation is actually better than the one of RAFT-Stereo.

To evaluate possible discrepancies a set of metrics inspired by Middlebury 2014, made up by the bad-τ percentage and the endpoint error (EPE), has been employed [34, 56], along with ℓ_1 maps between the two disparity maps, to localize the position of such discrepancies. The lighter areas of these maps represent points with an higher discrepancy. In general the discrepancies seems to be higher on the edges, and this is coherent to the fact that RAFT-Stereo tends to produce smoother transitions between depth levels, with respect to NeRF. The pairs of disparity maps with an higher EPE have been checked, finding out that the corresponding ℓ_1 maps contain, indeed, wider lighter areas.

In Figure 4.10 is presented an example where the arm of the Mario action figure is incorrectly reconstructed by RAFT-Stereo but correctly inferred by the NeRF model, which was supervised by RAFT-Stereo priors.

In Figure 4.11 is shown an instance in which the glass on the left is erroneously predicted by RAFT-Stereo, while the NeRF model is able to infer it. Moreover, the occlusion present in the raised arm of the frog is better captured by NeRF.

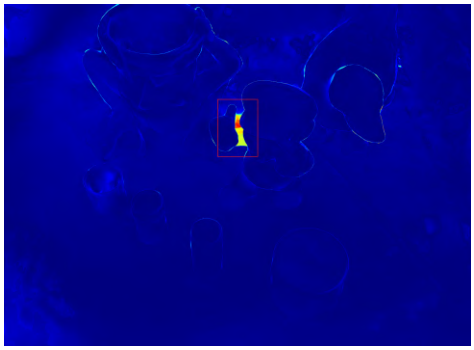
Even though NeRF seems to be able recover the cases of failure of RAFT-Stereo, the latter tends to produce smoother disparity maps, as depicted in Figure 4.12. However, these artifacts introduced by NeRF could be in principle filtered away or reduced by employing more views for the training.

In Table 4.1, the metrics of the previously presented examples are shown.

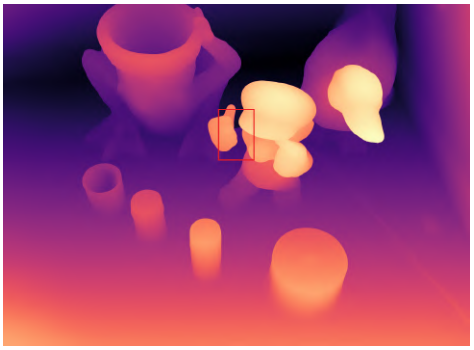
Example	EPE	bad-2	bad-4	bad-6	bad-8
Arm	1.46	0.16	0.06	0.03	0.01
Glass	1.66	0.15	0.05	0.02	0.01
Smoothness	1.66	0.23	0.08	0.03	0.01

Table 4.1: Metrics for examples evaluation.

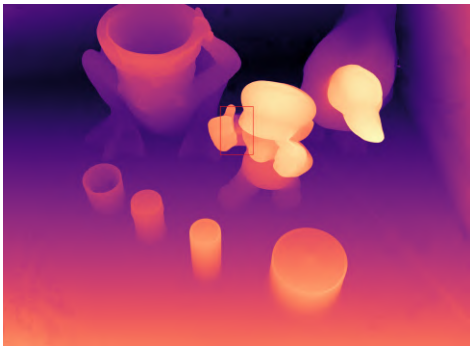
Essentially, it seems that NeRF is able to provide more informative depth maps, with respect to the priors on which it was trained, probably due to its multi-view nature. Nonetheless, these maps are not perfect, hence cleaning,



(a) ℓ_1 map.



(b) RAFT-Stereo disparity map.



(c) NeRF disparity map.

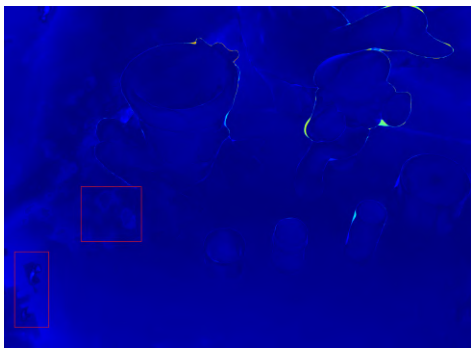
Figure 4.10: The discontinuity on the arm is localizable on the ℓ_1 map by checking the same bright area also in the disparity maps. It is clear that NeRF has been able to reconstruct properly the arm of the action figure, contrary to RAFT-Stereo.

(a) ℓ_1 map.

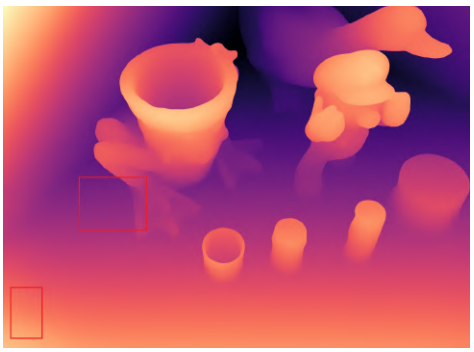
(b) RAFT-Stereo disparity map.

(c) NeRF disparity map.

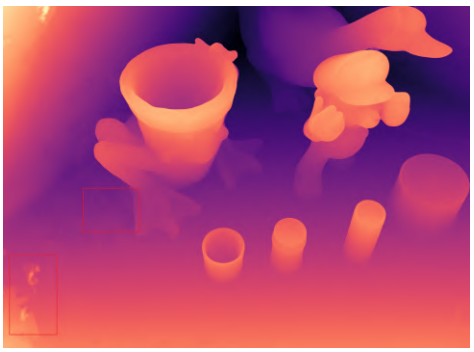
Figure 4.11: Also in this case is evident that NeRF has been able to reconstruct entirely the glass on the left, while RAFT-Stereo has not.



(a) ℓ_1 map.



(b) RAFT-Stereo disparity map.



(c) NeRF disparity map.

Figure 4.12: Even though the ℓ_1 map does not exhibit any major bright area it is noticeable that there are some artifacts on the disparity map generated by NeRF, especially on the borders of the image.

filtering and consistency checks are needed in order to enhance the overall quality. Also, producing more robust priors, by employing for instance a space-time framework, could aid the depth supervision of the NeRF model.

Conclusion

This thesis work aimed to mitigate two of the current main problems of stereo matching, namely finding pixel correspondences in presence of non-Lambertian surfaces and processing high-resolution images. Based on the performed experiments, the main objectives of the project have been addressed.

In particular, the project has improved the quality of the underlying depth maps associated with NeRF-based models by supervising the training with depth priors, generated by pre-trained models, achieving more coherent depth maps with respect to the employed depth priors. As apposed to other approaches, which exploit depth supervision to enhance the RGB reconstruction with fewer samples, the proposed methodology maintains an high number of samples along with the depth priors, during the training, to obtain high quality and sound depth maps in inference phase.

Furthermore, during the acquisition of the dataset, projectors have been employed to texturize the scene, in order to aid the matching process, while non-Lambertian surfaces have been covered with a vanishing scanning spray.

Finally, thanks to the novel view synthesis capabilities of Neural Radiance Fields, it has been possible to generate some multi-view datasets from the acquired scenes, which contains high-resolution images that present transparent objects, associated with satisfactory disparity maps.

For the scope of the thesis, these synthesis capabilities have been employed to infer high quality depth maps from each point of view of the acquired scene, and, in the case of the vanishing scanning spray, to generate the corresponding RGB view without the spray. In principle, it is possible to exploit these capabilities also to generate views unseen in the acquisition, along with the associated depth maps, simulating stereo pairs with different baselines. Moreover, models endowed with editing material and relighting capabilities, such as NeRFactor, could be exploited to change the appearance of already acquired scenes which does not present objects that exhibits non-Lambertian

behaviours, to insert synthetic but realistic non-Lambertian objects. Also, in order to make the usage of projectors more efficient, multiple images of the same view could be acquired in order to perform a space-time inference, achieving more robust depth priors.

In future, datasets produced with the described methodology could be employed as supervised data to fine-tune RAFT-Stereo and the other state-of-the-art deep networks, and, in general, data augmentation by means of Neural Radiance Fields could become a fundamental tool in Computer Vision.

Bibliography

[1] Filippo Aleotti et al. “Neural Disparity Refinement for Arbitrary Resolution Stereo”. In: (Oct. 2021).

[2] Jonathan T. Barron et al. “Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields”. In: *ICCV* (2021).

[3] Michael Bleyer and Sylvie Chambon. “Does Color Really Help in Dense Stereo Matching?” In: *In Proceedings of the international*. 2010.

[4] Piotr Bojanowski et al. “Optimizing the Latent Space of Generative Networks”. In: *ICML*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 599–608.

[5] Mark Boss et al. “NeRD: Neural Reflectance Decomposition from Image Collections”. In: *IEEE International Conference on Computer Vision (ICCV)*. 2021.

[6] Bernhard Buettgen et al. “CCD/CMOS lock-in pixel for range imaging: Challenges, limitations and state-of-the-art”. In: (Jan. 2005).

[7] Xiaotong Chen et al. “ClearPose: Large-scale Transparent Object Dataset and Benchmark”. In: *CoRR* abs/2203.03890 (2022).

[8] Xuelian Cheng et al. “Hierarchical Neural Architecture Search for Deep Stereo Matching”. In: *Advances in Neural Information Processing Systems* 33 (2020).

[9] James Davis, Ravi Ramamoorthi, and Szymon Rusinkiewicz. “Spacetime Stereo: A Unifying Framework for Depth from Triangulation”. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2003, pp. 359–366.

[10] Andrea Fusiello, Emanuele Trucco, and Alessandro Verri. “A compact algorithm for rectification of stereo pairs”. In: *Mach. Vis. Appl.* 12.1 (2000), pp. 16–22.

[11] Jason Geng. “Structured-light 3D surface imaging: a tutorial”. In: *Adv. Opt. Photon.* 3.2 (June 2011), pp. 128–160. DOI: 10.1364/AOP.3.000128. URL: http://opg.optica.org/aop/abstract.cfm?URI=aop-3-2-128.

[12] Yuan-Chen Guo et al. “NeRFReN: Neural Radiance Fields With Reflections”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 18409–18418.

[13] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2nd ed. Cambridge University Press, 2004. DOI: 10.1017/CBO9780511811685.

[14] Heiko Hirschmüller. “Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information”. In: *CVPR (2)*. IEEE Computer Society, 2005, pp. 807–814.

[15] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural Networks* 2.5 (1989), pp. 359–366.

[16] Lahav Lipson, Zachary Teed, and Jia Deng. “RAFT-Stereo: Multilevel Recurrent Field Transforms for Stereo Matching”. In: *International Conference on 3D Vision (3DV)*. 2021.

[17] Steven Liu et al. “Editing Conditional Radiance Fields”. In: *Proceedings of the International Conference on Computer Vision (ICCV)*. 2021.

[18] Yang Liu et al. “A Survey of Depth Estimation Based on Computer Vision”. In: *DSC*. IEEE, 2020, pp. 135–141.

[19] Charles T. Loop and Zhengyou Zhang. “Computing Rectifying Homographies for Stereo Vision”. In: *CVPR*. IEEE Computer Society, 1999, pp. 1125–1131.

[20] Wojciech Matusik et al. “A Data-Driven Reflectance Model”. In: *ACM Transactions on Graphics* 22.3 (July 2003), pp. 759–769.

[21] Moritz Menze, Christian Heipke, and Andreas Geiger. “Joint 3D Estimation of Vehicles and Scene Flow”. In: *ISPRS Workshop on Image Sequence Analysis (ISA)*. 2015.

[22] Ben Mildenhall et al. “NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis”. In: *ECCV*. 2020.

[23] Thomas Müller et al. “Instant Neural Graphics Primitives with a Multiresolution Hash Encoding”. In: *ACM Trans. Graph.* 41.4 (July 2022), 102:1–102:15. DOI: 10.1145/3528223.3530127. URL: https://doi.org/10.1145/3528223.3530127.

[24] Jacob Munkberg et al. “Extracting Triangular 3D Models, Materials, and Lighting From Images”. In: *arXiv:2111.12503* (2021).

[25] S.J.D. Prince. *Computer Vision: Models Learning and Inference*. Cambridge University Press, 2012.

[26] Nasim Rahaman et al. “On the Spectral Bias of Neural Networks”. In: *ICML*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 5301–5310.

[27] Christian Reiser et al. *KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs*. 2021. DOI: 10.48550/ARXIV.2103.13744. URL: https://arxiv.org/abs/2103.13744.

[28] Szymon Rusinkiewicz. “A New Change of Variables for Efficient BRDF Representation”. In: *Rendering Techniques*. Eurographics. Springer, 1998, pp. 11–22.

[29] Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach, Third International Edition*. Pearson Education, 2010.

[30] Shreeyak S. Sajjan et al. “ClearGrasp: 3D Shape Estimation of Transparent Objects for Manipulation”. In: *CoRR* abs/1910.02550 (2019).

[31] Daniel Scharstein and Richard Szeliski. “A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms”. In: *Int. J. Comput. Vis.* 47.1-3 (2002), pp. 7–42.

[32] Daniel Scharstein and Richard Szeliski. “High-Accuracy Stereo Depth Maps Using Structured Light”. In: *CVPR (1)*. IEEE Computer Society, 2003, pp. 195–202.

[33] Daniel Scharstein et al. “High-Resolution Stereo Datasets with Subpixel-Accurate Ground Truth”. In: *GCPR*. Vol. 8753. Lecture Notes in Computer Science. Springer, 2014, pp. 31–42.

[34] Daniel Scharstein et al. “High-Resolution Stereo Datasets with Subpixel-Accurate Ground Truth”. In: vol. 8753. Sept. 2014, pp. 31–42. ISBN: 978-3-319-11751-5. DOI: 10.1007/978-3-319-11752-2_3.

[35] Johannes Lutz Schönberger and Jan-Michael Frahm. “Structure-from-Motion Revisited”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[36] Johannes Lutz Schönberger et al. “Pixelwise View Selection for Unstructured Multi-View Stereo”. In: *European Conference on Computer Vision (ECCV)*. 2016.

[37] Tianchang Shen et al. “Deep Marching Tetrahedra: a Hybrid Representation for High-Resolution 3D Shape Synthesis”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[38] Zhelun Shen, Yuchao Dai, and Zhibo Rao. “CFNet: Cascade and Fused Cost Volume for Robust Stereo Matching”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 13906–13915.

[39] Robert Spangenberg, Tobias Langner, and Raúl Rojas. “Weighted Semi-Global Matching and Center-Symmetric Census Transform for Robust Driver Assistance”. In: *Computer Analysis of Images and Patterns*. Ed. by Richard Wilson et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 34–41. ISBN: 978-3-642-40246-3.

[40] Pratul P. Srinivasan et al. “NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis”. In: *CVPR*. 2021.

[41] Charles V. Stewart. “Robust Parameter Estimation in Computer Vision”. In: *SIAM Rev.* 41.3 (1999), pp. 513–537.

[42] Richard Szeliski. *Computer Vision - Algorithms and Applications, Second Edition*. Texts in Computer Science. Springer, 2022.

[43] Towaki Takikawa et al. “Neural Geometric Level of Detail: Real-Time Rendering With Implicit 3D Shapes”. In: *CVPR*. Computer Vision Foundation / IEEE, 2021, pp. 11358–11367.

[44] Matthew Tancik et al. “Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains”. In: *NeurIPS* (2020).

[45] Zachary Teed and Jia Deng. “RAFT: Recurrent All-Pairs Field Transforms for Optical Flow (Extended Abstract)”. In: *IJCAI*. ijcai.org, 2021, pp. 4839–4843.

[46] Matthias Teschner et al. “Optimized Spatial Hashing for Collision Detection of Deformable Objects”. In: *VMV*. Aka GmbH, 2003, pp. 47–54.

[47] Federico Tombari et al. “Classification and evaluation of cost aggregation methods for stereo correspondence”. In: *CVPR*. IEEE Computer Society, 2008.

[48] Fabio Tosi et al. “SMD-Nets: Stereo Mixture Density Networks”. In: *CVPR*. Computer Vision Foundation / IEEE, 2021, pp. 8942–8952.

[49] Vaibhav Vaish et al. “Reconstructing Occluded Surfaces Using Synthetic Apertures: Stereo, Focus and Robust Measures”. In: *CVPR (2)*. IEEE Computer Society, 2006, pp. 2331–2338.

[50] Igor Vasiljevic et al. “DIODE: A Dense Indoor and Outdoor DEpth Dataset”. In: *CoRR* abs/1908.00463 (2019).

[51] Dor Verbin et al. “Ref-NeRF: Structured View-Dependent Appearance for Neural Radiance Fields”. In: *CVPR* (2022).

[52] Viny Saajan Victor and Peter Neigel. *Survey on Semantic Stereo Matching / Semantic Depth Estimation*. 2021. DOI: 10.48550/ARXIV.2109.10123. URL: https://arxiv.org/abs/2109.10123.

[53] Kwang Hee Won and Soon Ki Jung. “hSGM: Hierarchical Pyramid Based Stereo Matching Algorithm”. In: *ACIVS*. Vol. 6915. Lecture Notes in Computer Science. Springer, 2011, pp. 693–701.

[54] Gengshan Yang et al. “Hierarchical deep stereo matching on high-resolution images”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5515–5524.

[55] Tao Yang et al. “3D ToF LiDAR in Mobile Robotics: A Review”. In: *CoRR* abs/2202.11025 (2022).

[56] Pierluigi Zama Ramirez et al. “Open Challenges in Deep Stereo: the Booster Dataset”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. CVPR. 2022.

[57] Xiuming Zhang et al. “NeRFactor: Neural Factorization of Shape and Reflectance under an Unknown Illumination”. In: *ACM Trans. Graph.* 40.6 (Dec. 2021). ISSN: 0730-0301. DOI: 10.1145/3478513.3480496. URL: https://doi.org/10.1145/3478513.3480496.