SR-MVS:
Multi-View Stereo enhancement through
Super-Resolution

Supervisor: Prof. Matteo Matteucci
Co-Supervisor: Eng. Andrea Romanoni

Master Thesis by:
Eugenio Lomurno
Matricola 883083

Academic Year 2018-2019
To my loved family
“Thoughts without content are empty, intuitions without concepts are blind.”
Immanuel Kant, *Critique of Pure Reason*
Abstract

Nowadays, 3D reconstruction algorithms have reached incredible levels of accuracy and robustness, especially when the images of the environment to be reconstructed are equipped with a high resolution. There are cases in which, unfortunately, the quality of these images is not high enough, as in the case of old photographs, or cases in which the hardware used to immortalize them is a limitation.

In this thesis, we discuss a new way to improve the quality of 3D reconstructions performed starting from a set of images of the scene to be modelled. More precisely, we will examine whether the increase in the resolution of these input images results in an improvement of the reconstructed model. In particular, we study whether, how and how much the depth map estimation, being the most relevant step of the most successful Multi-View Stereo pipelines, can benefit from Super-Resolution based on Deep Learning algorithms. Although these latter techniques can generate artifacts, we demonstrate how their application before restoring the depth maps leads to the realization of a better 3D model both in the case of algorithms based on PatchMatch and in those based on autonomous learning.
Summary

Al giorno d’oggi, gli algoritmi di ricostruzione 3D hanno raggiunto incredibili livelli di accuratezza e di robustezza, soprattutto quando le immagini dell’ambiente da ricostruire sono dotate di una risoluzione elevata. Esistono casi in cui, sfortunatamente, la qualità di queste immagini non è sufficientemente alta, come nel caso di vecchie fotografie, o casi in cui l’hardware utilizzato per immortalarle rappresenta una limitazione.

Acknowledgments

Ci tengo a ringraziare in maniera particolare il Professor Matteuci, non solo per il tempo dedicatomi e per i preziosi consigli, ma soprattutto per essere una grande persona e fonte inesauribile di idee.

I miei più sentiti ringraziamenti vanno ad Andrea, per avermi affiancato passo dopo passo in questo percorso e per essere apparso ai miei occhi come amico oltre che come guida. Ringrazio Marco per la sua preziosa pazienza, la sua invidiabile preparazione e per la sua gentilezza.

Ringrazio i miei compagni di team e di laboratorio, nonché amici, Giovanni, Chiara e Refik, con i quali ho condiviso fatiche e grandi traguardi. Un ringraziamento sentito va a tutte le altre meravigliose persone con le quali ho percorso questa esperienza universitaria, in particolare a Burcu, Giacomo F, Giacomo B ed Emilia.

Ci tengo a ringraziare particolarmente Riccardo per essersi fatto in quattro per me ed avermi illuminato con la sua ironia e intelligenza, Nicola per avermi motivato e spinto a mettermi in gioco, Alessandro per avermi fatto a più riprese riscoprire la semplicità ed Eliana per avere sempre creduto in me.

Ringrazio i miei cari amici di Matera, in particolare Domenico P, Gianvito, Leonardo, Gianfranco, Annasofia, Alessandra, Beatrice, Davide, Domenico D e Arturo per essermi stati accanto nonostante la distanza. Ringrazio i miei coinquilini Martino e Leonardo, per l'affetto dimostratomi e per avermi supportato in questo momento particolare della vita di tutti.

Infine, un immenso ringraziamento va alla mia famiglia per avermi capito, sostenuto ed incoraggiato in tutti questi anni e per aver reso possibile tutto ciò. Mi reputo fortunato e sono fiero di poter dire che conserverò per sempre un bel ricordo di questo lungo viaggio.

A voi tutti, grazie.
# Contents

Abstract ............................... II

Summary ................................ IV

Acknowledgments ....................... VI

1 Introduction .......................... 1

2 Background Knowledge ............... 5
   2.1 Deep Learning ........................ 7
      2.1.1 History .......................... 8
      2.1.2 From Perceptron to Feedforward Neural Networks 10
      2.1.3 Loss Functions .................... 13
      2.1.4 Backpropagation .................. 16
      2.1.5 Convolutional Neural Networks 21
      2.1.6 Recurrent Neural Networks ........ 24
   2.2 Camera Geometry .................... 28
      2.2.1 Image Formation ................... 29
      2.2.2 Epipolar Geometry ................ 31
      2.2.3 Triangulation ..................... 33
      2.2.4 Multi-View Geometry .............. 34
   2.3 Sparse Reconstruction ............... 36

3 State of the Art ....................... 39
   3.1 Single-Image Super-Resolution ....... 40
      3.1.1 Problem Formulation ............... 41
      3.1.2 Classical Super-Resolution ........ 44
      3.1.3 Deep Learning-based Super-Resolution .... 46
      3.1.4 Deep Back-Projection Network ....... 50
   3.2 Multi-View Stereo .................... 53
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.1 Image Appearance</td>
<td>54</td>
</tr>
<tr>
<td>3.2.2 Photo-Consistency</td>
<td>55</td>
</tr>
<tr>
<td>3.2.3 Depth Maps</td>
<td>57</td>
</tr>
<tr>
<td>3.2.4 PatchMatch-based and Learning-based MVS</td>
<td>59</td>
</tr>
<tr>
<td>3.2.5 COLMAP</td>
<td>61</td>
</tr>
<tr>
<td>3.2.6 TAPA-MVS</td>
<td>63</td>
</tr>
<tr>
<td>3.2.7 CasMVSNet</td>
<td>67</td>
</tr>
<tr>
<td>4 Super-Resolution Multi-View Stereo</td>
<td>71</td>
</tr>
<tr>
<td>5 Experiments and Results</td>
<td>77</td>
</tr>
<tr>
<td>5.1 COLMAP-SR</td>
<td>79</td>
</tr>
<tr>
<td>5.1.1 ETH3D train</td>
<td>79</td>
</tr>
<tr>
<td>5.1.2 ETH3D test</td>
<td>85</td>
</tr>
<tr>
<td>5.2 TAPA-MVS-SR</td>
<td>87</td>
</tr>
<tr>
<td>5.2.1 ETH3D train</td>
<td>87</td>
</tr>
<tr>
<td>5.2.2 ETH3D test</td>
<td>90</td>
</tr>
<tr>
<td>5.3 CasMSVNet-SR</td>
<td>93</td>
</tr>
<tr>
<td>5.3.1 ETH3D train</td>
<td>93</td>
</tr>
<tr>
<td>5.3.2 ETH3D test</td>
<td>96</td>
</tr>
<tr>
<td>5.4 Final Results</td>
<td>98</td>
</tr>
<tr>
<td>6 Conclusions and Future Works</td>
<td>105</td>
</tr>
<tr>
<td>Bibliography</td>
<td>109</td>
</tr>
</tbody>
</table>

Appendix A Reconstructions
List of Figures

1.1 Example of 3D reconstruction through MVS. . . . . . . 4
2.1 Example of a 3D sparse reconstruction of central Rome. 5
2.2 Summary of Artificial Intelligence, Machine Learning and Deep Learning differences. . . . . . . . . . . . . . . 7
2.3 General scheme of a biological neuron architecture. . . 10
2.4 General scheme of a Perceptron architecture . . . . . . 11
2.5 General scheme of a Feedforward Neural Network architecture. . . . . . . . . . . . . . . . . . . . . 13
2.6 Graphical representation of a loss function hyperplane. 16
2.7 Scheme of the simplest Feedforward Neural Network architecture. . . . . . . . . . . . . . . . . . . . . 18
2.8 Scheme of the convolution operator used in CNNs. . . . 22
2.9 Scheme of the pooling operator used in CNNs. . . . . . 23
2.10 An unrolled Recurrent Neural Network. . . . . . . . . 25
2.11 An unrolled Long-Short Term Memory. . . . . . . . . 26
2.12 Basic model of a single camera. . . . . . . . . . . . . . 28
2.13 Scheme of the Pinhole camera model . . . . . . . . . . 30
2.14 Scheme of epipolar geometry model . . . . . . . . . . . 32
2.15 Scheme of Structure from Motion’s error reprojection. . 37
3.1 Example of Single-Image Super-Resolution on Pulse cover 40
3.2 Example of Single-Image Super-Resolution on Pulse cover (details) . . . . . . . . . . . . . . . . . . . . . 43
3.3 Chart of SRCNN performance . . . . . . . . . . . . . . 47
3.4 SISR comparison over Butterfly . . . . . . . . . . . . . . 48
3.5 Super-Resolution with scale factor x8 comparison . . . 50
3.6 Architecture of basic Deep Back-Projection Network . . 51
3.7 3D dense reconstruction of the Louvre museum made through the use of COLMAP . . . . . . . . . . . . . . . 53
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.8</td>
<td>Scheme of Lambertian model</td>
<td>55</td>
</tr>
<tr>
<td>3.9</td>
<td>Depth estimation from camera $C_1$ of pixel $x_i$ w.r.t camera $C_2$</td>
<td>57</td>
</tr>
<tr>
<td>3.10</td>
<td>The general PatchMatch scheme</td>
<td>59</td>
</tr>
<tr>
<td>3.11</td>
<td>3D dense reconstruction of the Todai-ji temple made through the use of COLMAP</td>
<td>62</td>
</tr>
<tr>
<td>3.12</td>
<td>Example of TAPA-MVS filtering</td>
<td>64</td>
</tr>
<tr>
<td>3.13</td>
<td>Example of TAPA-MVS depth map refinement pipeline</td>
<td>66</td>
</tr>
<tr>
<td>3.14</td>
<td>Graphical representation of CasMVSNet architecture</td>
<td>68</td>
</tr>
<tr>
<td>4.1</td>
<td>Scheme of Super-Resolution Multi-View Stereo pipeline</td>
<td>73</td>
</tr>
<tr>
<td>5.1</td>
<td>Charts of COLMAP default model over ETH3D train set</td>
<td>80</td>
</tr>
<tr>
<td>5.2</td>
<td>COLMAP depth maps filtered through filterSpeckles</td>
<td>83</td>
</tr>
<tr>
<td>5.3</td>
<td>Charts of COLMAP model over ETH3D train set</td>
<td>84</td>
</tr>
<tr>
<td>5.4</td>
<td>Charts of COLMAP model over ETH3D test set</td>
<td>86</td>
</tr>
<tr>
<td>5.5</td>
<td>Charts of TAPA-MVS model over ETH3D train set</td>
<td>89</td>
</tr>
<tr>
<td>5.6</td>
<td>TAPA-MVS Textureless depth maps comparison</td>
<td>91</td>
</tr>
<tr>
<td>5.7</td>
<td>Charts of TAPA-MVS model over ETH3D test set</td>
<td>92</td>
</tr>
<tr>
<td>5.8</td>
<td>CasMVSNet and CasMVSNet-SR comparison over electro dataset</td>
<td>94</td>
</tr>
<tr>
<td>5.9</td>
<td>Charts of CasMVSNet model over ETH3D train set</td>
<td>95</td>
</tr>
<tr>
<td>5.10</td>
<td>Charts of CasMVSNet model over ETH3D test set</td>
<td>97</td>
</tr>
<tr>
<td>5.11</td>
<td>Pipelines comparison over ETH3D Overall sequences</td>
<td>99</td>
</tr>
<tr>
<td>5.12</td>
<td>Pipelines comparison over ETH3D Textureless sequences</td>
<td>101</td>
</tr>
<tr>
<td>5.13</td>
<td>Pipelines comparison over ETH3D Textured sequences</td>
<td>103</td>
</tr>
</tbody>
</table>
List of Tables

3.1 Deep Learning-based SISR comparison ........................................ 49

5.1 F1 scores of COLMAP default model over ETH3D train set .......................... 81
5.2 F1 score comparison between different base-resolution COLMAP models over ETH3D train set .................................................. 82
5.3 F1 score comparison between different Super-Resolution COLMAP models over ETH3D train set ............................................... 82
5.4 F1 scores of COLMAP model over ETH3D train set ............................... 83
5.5 F1 scores of COLMAP model over ETH3D test set ................................. 85
5.6 F1 scores of TAPA-MVS model over ETH3D train set ............................. 88
5.7 F1 scores of TAPA-MVS model over ETH3D test set ............................... 90
5.8 F1 scores of CasMVSNet model over ETH3D train set ............................ 94
5.9 F1 scores of CasMVSNet model over ETH3D test set ............................. 96
5.10 Pipelines improvement comparison over ETH3D Overall sequences ............................................................. 98
5.11 Pipelines improvement comparison over ETH3D Textureless sequences .................................................. 100
5.12 Pipelines improvement comparison over ETH3D Textured sequences ................................................. 102
5.13 F1, accuracy and completeness final scores over ETH3D low-resolution multi-view benchmark ................................................ 104
Chapter 1

Introduction

The reconstruction of a 3D model of a scene starting from images that immortalize it is a fascinating and complex problem on which many researchers and scientists have concentrated their efforts. The purposes of this task are varied, and are fundamental in many scenarios, for example during the mapping of large cities, in the conservation of archaeological heritage, in the new and stimulating world of autonomous driving, in the field of robot localization or in the reconstruction of organs and apparatus for clinical purposes.

In the world of the Computer Vision scientific community, this type of problem is called Multi-View Stereo (MVS) and has as its objective the reconstruction of the most robust and complete 3D model possible. An example of reconstruction performed with this type of algorithm is shown in Figure 1.1. Currently, the most successful workflow for performing such reconstructions is as follows. First, a Structure from Motion algorithm estimates camera parameters, such as positions and orientations. Then follows a phase of estimating the depth map. The most common approach is based on the PatchMatch model pioneered by Bleyer et al. (2011), however recent autonomous Learning-based algorithms have shown promising results. On the one hand, the first approach leads to very precise results, while on the other it produces more complete models, even if it presents scalability problems. Finally, the depth maps are projected on three dimensions and the point clouds are joined together.

In controlled scenarios, where the hardware used to collect images is not subject to particular constraints, it is relatively easy to acquire high-resolution images and obtain a high-quality reconstruction of the
scene following the previous pipeline. However, in several cases, the input of an MVS method consists of low-resolution images. For example, when power consumption limits hardware, e.g., with drones or during the processing of images shot by satellites or during the recovery of a 3D model from old photos. In such cases, regardless of the MVS algorithm adopted, the recovered 3D model will probably be incomplete or lacking in detail. Increasing the resolution of the input image is the most direct attempt to overcome such a problem.

In this regard, another branch of the Computer Vision community has studied the problem of improving the quality of images by enriching them with details. This type of activity is called Super-Resolution and in recent years has exploited the most recent and accurate techniques based on artificial neural networks, managing to obtain excellent results, producing less and less artifacts over time.

In this thesis, we aim to investigate whether, how and how much the newer and more powerful versions of these algorithms can improve the quality of 3D models reconstructed through MVS. We take advantage of the Super-Resolution’s ability to infer details at higher resolutions by applying it directly to the input images, before estimating the depth maps. At the time of writing, as far as we know, this turns out to be the first scientific study of this type of approach. We are therefore motivated to give our contribution to the Computer Vision community, generalizing our results so that they can have the robustness and accuracy necessary to ensure the emerged improvements can be replicated and adapted over the models of anyone who wishes.

This document is structured as follows:

- In Chapter 2 we provide a theoretical background on Deep Learning algorithms, starting from basic neural models up to introducing more complex architectures, and on the techniques known as Camera Geometry, used to project objects from the three-dimensional world to the two-dimensional plane.

- In Chapter 3 we present both the necessary background and related works in the fields of Single-Image Super-Resolution and Multi-View Stereo. We also describe the models intended for the part of experiments, that is Deep Back-Projection Network by Haris et al. (2019) as regards Super-Resolution, and COLMAP by Schönberger et al. (2016), TAPA-MVS by Romanoni and Mat-
teucci (2019) as well as CasMVSNet by Gu et al. (2019) as regards the part of Multi-View Stereo.

- In Chapter 4 we describe the characteristics of the pipeline proposed for each Multi-View Stereo model, justifying the choices and parameters used.

- In Chapter 5 we present the publicly available datasets that we used in our analysis. Moreover, we show a wide range of experimental results and comparisons between the basic models and our pipelines, providing a detailed description for each of these.

- Finally, in the last chapter, we resume our work and discuss some possible future developments and improvements.
Chapter 1. Introduction

Figure 1.1: Example of 3D reconstruction through a Multi-View Stereo pipeline. In this case, the reconstructed scene is named Ignatius, from Tanks and Temples benchmark by Knapitsch et al. (2017), and the algorithm used is named COLMAP by Schönberger et al. (2016)
Chapter 2

Background Knowledge

3D reconstruction as well as Super-Resolution are two topics belonging to the world of Computer Vision which are based on very complex mathematical models and algorithms that have required the efforts of various scientists and researchers to be developed. Given their complexity, in this chapter, we will provide the basic knowledge to be able to understand the mechanisms that underlie them and to provide the tools necessary to understand the contents proposed in the following chapters of this thesis.

In more detail, we will start by talking about the fundamental elements in the field of Artificial Intelligence based on neural networks, known as Deep Learning, explaining how these technologies were born and evolved during the last century, to become those algorithms of our days so famous and at the same time with unknown behaviour. Subsequently, we will also give a general intuition relating to the field of Computer Vision known as Camera Geometry, describing the basic

\[ \text{Figure 2.1: 3D sparse reconstruction of central Rome realized through the use of the Structure from Motion algorithm of COLMAP (From Schönberger and Frahm (2016)).} \]
Chapter 2. Background Knowledge

techniques on which the models we will use are heavily based.

We will give an intuition on how modern algorithms manage to project an object belonging to the three-dimensional world on a two-dimensional plane starting from one or more cameras through the use of parameters and matrices, up to introducing the most modern and complex Structure from Motion concept, able to reconstruct sparse representations of scenes like the one shown in Figure 2.1.
2.1 Deep Learning

In computer science, *Artificial Intelligence* is the field that studies if, how and how much machines are able to demonstrate intelligence, in contrast to the natural intelligence displayed by human beings. It includes many areas, but the most popular and widely used is *Machine Learning*. It is known as the scientific study of statistical models and algorithms used by computer systems to execute specific tasks without being explicitly programmed with instructions, but relying on complex patterns inside data and statistical inference instead. During the last decade, we have witnessed to the rise of *Artificial Neural Networks* and *Deep Learning*, black-box algorithms able to solve a wide pool of tasks and to outperform human abilities in many of them. Deep Learning is actually the most investigated and promising field in Machine Learning Figure 2.2, and can be defined as a family of architectures, the Artificial Neural Networks, that uses multiple sets of logical units, called artificial neurons, grouped in layers to progressively extract higher-level information from the raw input data.

In this section we will briefly cross the history of Deep Learning, we will see in which way its architectures are able to adaptively learn to successfully execute simple and complex tasks. Then, we will understand which are the main layer’s typologies used in Computer Vision, and give an overview of the architectures that have made Deep Learning powerful as it is nowadays.

*Figure 2.2: Summary of Artificial Intelligence, Machine Learning and Deep Learning differences.*
2.1.1 History

The roots of modern Deep Learning (DL) can be traced back to the year in which McCulloch and Pitts (1943) created a model based on biological neural networks with the aim of simulating a thought process. This can be formally considered as the first example of an artificial neuron. Then after a few years, Hebb (1949) described a theory on behaviour, based on the physiology of the nervous system which will go down in history as Hebbian Learning. It is the first step to give artificial neurons the ability to learn.

Subsequently, Turing (1950) proposed a theory on a machine to determine whether a computer can "think" or not. He crafted what has been called The Turing Test, although he himself called it The Imitation Game. It was a test that included the presence of a man A, a woman B and a third person C. The latter is kept separate from the other two and has the task of establishing through a series of questions who is the man and who the woman. In this game, A is in charge of deceiving C, while B is in charge of helping him. To ensure that C cannot have further addresses, the answers to his questions are printed and transmitted in the same way. The test is based on replacing A with a machine. If the percentage by which C manages to guess is similar before and after this replacement, then the machine is said to have passed the test. It would take 60 years for any machine to do so, although many still debate the validity of the results.

Finally, thanks to the effort of Rosenblatt (1958), a psychologist, the first physical neural computing model was born, and it has been called Perceptron. Even if his idea was more based on a hardware architecture than on software or an algorithm, it planted the seeds of bottom-up learning and is widely recognized as the foundation of Artificial Neural Networks (ANN). Two years later Kelley (1960) developed the basics of a continuous backpropagation model, an algorithm that will be widely used to train ANN after some decade. While this concept of backpropagation, the backward propagation of errors for purposes of training, did exist in the early 1960s, it was clumsy and inefficient, and would not become useful until 1985. Then Ivakhnenko (1965) created the first working DL networks, applying what had been only theories and ideas since that moment. For that reason alone, Ivakhnenko is considered by many as the father of modern DL.
During the 1970’s the first Artificial Intelligence (AI) winter kicked in, the result of promises that could not be kept due to two very basic limitations: not enough memory and not enough processing speed. The impact of this lack of funding has had a very negative impact on AI research. At the end of this period, Fukushima (1979), a computer science researcher, created the father of nowadays Convolutional Neural Networks, an ANN that learned how to recognize visual patterns called Neocognitron. A few years later, Hopfield (1982) created and popularized a system introducing the concept of modern Recurrent Neural Networks (RNN), architectures where connections between artificial neurons form a directed graph along a temporal sequence.

These two last contributes have been followed by the second period of decreasing interest, that goes under the name of second AI winter, caused by an excessive optimism from investors that, from a certain point on, decided to stop funding AI projects and researches. During this period, Rumelhart et al. (1986) wrote a paper in which described in greater detail the process of backpropagation. They showed how it could vastly improve the existing neural networks for many tasks such as shape recognition, word prediction, and more. Finally, the 1990s have seen the rise of DL thanks to many brilliant models, architectures and algorithms that will be widely used in 21st century, able to solve complex tasks and to show the strength of ANN. One of the most important researchers that must be mentioned is LeCun et al. (1989), a star in AI and DL world, that with his research team was able to combine Convolutional neural networks, a technology still in developing phase, with recent backpropagation theories to read handwritten digits. Another huge contribute has been given by Hochreiter and Schmidhuber (1997) with their innovative RNN called Long Short-Term Memory, a model able to "remember" information too far back in temporal sequences, that solved a problem named Vanishing Gradient. Again, LeCun et al. (1998) developed the stochastic gradient descent algorithm combined with backpropagation.

From the 2000s to nowadays, and especially after the hardware performance improvement of 2010s, DL has reached levels of complexity incomparable with aforementioned works, and it is still keeping growing thanks to the attention it has received from Hi-tech giants such as Google and Facebook. In this period we saw the creation of important datasets for training purposes proposed by Deng et al. (2009), the birth
of Deep CNN architectures inspired to Krizhevsky et al. (2012) work, and of complex models like Variational Auto-Encoders by Kingma and Welling (2013) or Generative Adversarial Networks by Goodfellow et al. (2014). All of this let DL be so much stronger to overtake human being abilities in many tasks, such as in faces and object recognition or playing ATARI games, but also to generate audio and visual art or translate a text in real-time.

2.1.2 From Perceptron to Feedforward Neural Networks

The *Artificial Neuron*, also named *Perceptron*, is a mathematical model that aims to simulate the behaviour of a biological neuron. From biology and neuroscience, we know that there are various types and families of neurons, but it is generally correct to say that a neuron is a cell belonging to the nervous system and typically composed by a single cell body, called soma, by many dendrites, and by a single axon as shown in Figure 2.3. It receives electrical signals through the dendrites, which forward them to the soma, which in turn repropagates them through the axon to other neurons, if and only if this received impulse is sufficient to overcome an excitation threshold.

Starting from this knowledge, the Perceptron is a model composed of many inputs, a computational unit and a single output. At the same way of a natural neuron, it is able to forward information if a chosen prefixed threshold is exceeded. In details, the Perceptron is a learning algorithm which falls into the family of binary classifiers, characterized

![General scheme of a biological neuron architecture.](image)

*Figure 2.3: General scheme of a biological neuron architecture.*
by an input vector \( \mathbf{x} \) of \( N \) real values, a set of \( N \) real values \( \mathbf{w} \) called weights, one for each input, an activation or threshold function \( \phi() \), and a single binary output \( \hat{f}(\mathbf{x}) \), as shown in Figure 2.4, so that:

\[
\hat{f}(\mathbf{x}) = \begin{cases} 
1 & \text{if } \phi(\mathbf{w} \cdot \mathbf{x} + b) > 0 \\
0 & \text{otherwise}
\end{cases}
\] (2.1)

having that \( \phi() \) is the Heaviside function:

\[
\phi(\mathbf{x}) = \begin{cases} 
1 & \text{if } \mathbf{x} > 0 \\
0 & \text{otherwise}
\end{cases}
\] (2.2)

and it can also be replaced with other functions, e.g. sign function or linear function, \( b \) is the bias and \( \mathbf{w} \cdot \mathbf{x} \) is the dot product defined as \( \sum_{n=1}^{N} w_n \cdot x_n \). Putting all together, we obtain the following model:

\[
\hat{f}(\mathbf{x}) = \phi(\mathbf{w} | \mathbf{x}, b) = \begin{cases} 
1 & \text{if } \phi(\sum_{n=1}^{N} w_n \cdot x_n + b) > 0 \\
0 & \text{otherwise}
\end{cases}
\] (2.3)

The outcome of this binary classifier is used to separate \( \mathbf{x} \) in two different classes. From a geometric point of view, this output is named decision boundary, and while the dot product has the utility of rotate it with respect to the \( x \) axis, the bias has the utility of move it along the \( y \) axis. The problem to solve consists in finding the optimal weights, and the iterative learning algorithm used to update them is called Hebbian learning rule.
Its application consists of initializing all the weight, to zero or a negative value, and then iteratively updating them for each miss-classification so that:

$$w^{t+1} \leftarrow w^t + \Delta w^t$$

$$\Delta w^t = \eta (f(x) - \hat{f}(x)^t)x$$

(2.4)

where $t$ is a positive integer representing the current iteration, $\eta$ is a positive real number named learning rate and $f(x)$ is the vector containing the correct labels associated to $x$. It is proven that thanks to this algorithm, the Perceptron is able to solve any problem of linear binary classification in a finite number of weight updates. In case of non-linear input data, i.e. $x$ samples cannot be correctly separated according to $f(x)$ in two regions by a single hyperplane, Hebbian learning rule applied to a Perceptron learning will completely fail and a stronger model is required to solve this kind of problems. If we use many stratified artificial neurons, we can obtain such a model.

A Feedforward Neural Network or Multilayer Perceptron is a non-linear model composed of one or more layers of Perceptrons, each one with its weights and activation function. It is the first and simplest typology of Artificial Neural Network, in which each artificial neuron forwards its output to each neuron of the next layer in only one direction, i.e. from the so-called input layer to the output layer, avoiding loops or cycles inside the whole network.

As shown in Figure 2.5, between the aforementioned layers there may be one or more hidden layers that works exactly like the others, with the difference that they do not receive directly raw input data and forwards their output only to other layers, i.e. not like the output layer. Unlike Perceptron, a Feedforward Neural Network is a model able to solve a wide set of tasks, e.g. multilabel classification, regression and ranking, by exploiting the ability of hidden layers to create a high-level representation from raw input data that is adaptively adjusted in order to produce accurate outputs. For this family of architectures, the Hebbian learning rule is no longer able to correctly update the weights, so it requires a different and more complex iterative algorithm to solve this task. Before presenting such a procedure, we need to introduce the concept of loss function.
2.1.3 Loss Functions

To understand how much an Artificial Neural Network is doing well or bad in a task, we need to measure its performance. For simplicity, we will refer to a model with $I$ input neurons, a single hidden layer with $J$ neurons and a single neuron in output layer without considering the bias, exactly like in Figure 2.7, but the following reasoning can be extended to each Feedforward Neural Network architecture. We can describe the model output $\hat{y}$ as:

$$\hat{y} = \phi(x|W, w) = \phi\left(\sum_{j=1}^{J} W_j \cdot \rho\left(\sum_{i=1}^{I} w_{ji} \cdot x_i\right)\right)$$

where $\phi$ is the activation function of the output layer, $W_j$ is the weight between the $j^{th}$ neuron of the hidden layer and the output neuron, $\rho$ is the activation function of the hidden layer and $w_{ji}$ is the weight between
the $i^{th}$ input neuron and the $j^{th}$ neuron of the hidden layer. Given a dataset $\mathbf{D}$ with $N$ pairs of features and targets $(\mathbf{x}, \mathbf{y})$, training an Artificial Neural Network means minimizing the average error between $\mathbf{y}$ and the produced predictions $\hat{\mathbf{y}}$ given the observation of $\mathbf{x}$. Assuming that these observations are sampled from a Gaussian distribution with known variance $\sigma^2$:

$$x_1, x_2, ..., x_N \sim \mathcal{N}(\mu, \sigma^2)$$

$$p(\mathbf{x} | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

(2.6)

and that the generic $n^{th}$ model prediction $\hat{\mathbf{y}}_n$ as:

$$\hat{\mathbf{y}}_n = \phi(\mathbf{x}_n | W, w) = \phi(\sum_{j=1}^{J} W_j \cdot \rho(\sum_{i=1}^{I} w_{ji} \cdot x_{i,n}))$$

(2.7)

according to $\mathbf{y}$, if it is a set of continuous values, we are facing a regression task, instead, if it is composed of discrete values, we are facing a classification problem. In both cases, we need to compute an adequate loss function to minimize the error, so we let $\theta = (\theta_1, \theta_2, ..., \theta_k)^T$ be a vector of parameters. Starting from $p(\text{Data} | \theta)$ we look for the Maximum Likelihood Estimation (MLE) for $\theta$ because it is equivalent to minimize the loss function.

In the case of regression, the goal is to approximate the target $\mathbf{y}$ having that:

$$y_n = \phi(x_n | W, w) + \epsilon_n, \epsilon_n \sim \mathcal{N}(0, \sigma^2) \Rightarrow y_n \sim \mathcal{N}(\phi(x_n | w), \sigma^2)$$

$$p(\mathbf{y} | \phi(\mathbf{x}|W, w), \sigma^2) = p(\mathbf{y} | \hat{\mathbf{y}}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(\mathbf{y}-\hat{\mathbf{y}})^2}{2\sigma^2}}$$

(2.8)

We write the likelihood function:

$$L(W, w) = P(\text{Data} | \theta) = p(y_1, y_2, ..., y_N | \phi(\mathbf{x}|W, w), \sigma^2)$$

$$= p(y_1, y_2, ..., y_N | \hat{\mathbf{y}}, \sigma^2) = \prod_{n=1}^{N} p(y_n | \hat{y}_n, \sigma^2)$$

$$= \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(y_n-\hat{y}_n)^2}{2\sigma^2}}$$

(2.9)
and we compute the weights which maximize this likelihood:

\[
\text{argmax}_{W,w} L(W, w) = \text{argmax}_{W,w} \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(y_n - \hat{y}_n)^2}{2\sigma^2}}
\]

\[
= \text{argmax}_{W,w} \sum_{n=1}^{N} \log\left( \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(y_n - \hat{y}_n)^2}{2\sigma^2}} \right)
\]

\[
= \text{argmax}_{W,w} \sum_{n=1}^{N} \left( \log\frac{1}{\sqrt{2\pi\sigma}} - \frac{1}{2\sigma^2}(y_n - \hat{y}_n)^2 \right)
\]

\[
= \text{argmin}_{W,w} \sum_{n=1}^{N} (y_n - \hat{y}_n)^2
\]

which is a commonly chosen loss function for regression problems named \textit{Residual Sum of Squares} (RSS).

In the case of binary classification, the goal is to approximate the posterior probability of \(y\) having that:

\[
\hat{y}_n = \phi(x_n|W, w) = p(y_n|x_n)
\]

\(y_n \in 0, 1 \Rightarrow y_n \sim Be(\phi(x_n|W, w))\)

\[
p(y|\phi(x|W, w)) = p(y|\hat{y}) = \hat{y}^{y} \cdot (1 - \hat{y})^{(1-y)}
\]

We write the likelihood function:

\[
L(W, w) = P(Data|\theta) = p(y_1, y_2, ..., y_N|\phi(x|W, w))
\]

\[
= p(y_1, y_2, ..., y_N|\hat{y}) = \prod_{n=1}^{N} p(y_n|\hat{y}_n)
\]

\[
= \prod_{n=1}^{N} \hat{y}_n^{y_n} \cdot (1 - \hat{y}_n)^{(1-y_n)}
\]

(2.12)
and we compute the weights which maximize this likelihood:

\[
\argmax_{W,w} L(W, w) = \argmax_{W,w} \prod_{n=1}^{N} \hat{y}_n^{y_n} \cdot (1 - \hat{y}_n)^{(1-y_n)}
\]

\[
= \argmax_{W,w} N \sum_{n=1}^{N} (y_n \log(\hat{y}_n) + (1 - y_n) \log(1 - \hat{y}_n)) \tag{2.13}
\]

\[
= \argmin_{W,w} - N \sum_{n=1}^{N} (y_n \log(\hat{y}_n) + (1 - y_n) \log(1 - \hat{y}_n))
\]

which is a commonly chosen loss function for binary classification problems named *Binary Crossentropy*.

### 2.1.4 Backpropagation

Now that we know how to compute a loss functions according to the chosen task, we can introduce the technique named *backpropagation*, the algorithm designed by LeCun et al. (1998) to use gradient methods to train Feedforward Neural Networks and update the weights to minimize the chosen loss function.

![Graphical representation of a loss function hyperplane. It is possible to observe the presence of many local minima values with respect to the weights.](image)

*Figure 2.6: Graphical representation of a loss function hyperplane. It is possible to observe the presence of many local minima values with respect to the weights.*
More in detail, gradient descent, as well as more sophisticated techniques such as stochastic gradient descent, or momentum methods, is applied over the loss function with respect to each weight, one layer at a time through the use of the chain rule, iterating backwards from the last to the first layer, avoiding redundancy in computations and making the whole algorithm scalable. Moreover, gradient descent is a first-order iterative optimization algorithm able to find local minima of differentiable functions, and in some cases even the global minimum.

Let \( w \) be the weight vector between two generic layers, \( E \) the chosen loss function and \( \eta \) the learning rate. For each iteration of backpropagation we have that:

\[
\begin{align*}
  w^{t+1} & \leftarrow w^t + \partial w^t \\
  \partial w^t & = -\eta \cdot \frac{\partial E}{\partial w}
\end{align*}
\]  

(2.14)

Referring to Equation (2.5) and to Figure 2.7:

\[
\begin{align*}
  a_j & = \sum_{i=1}^{I} w_{ji} \cdot x_i \\
  b_j & = \rho(a_j) \\
  A & = \sum_{j=1}^{J} W_j \cdot b_j \\
  \hat{y} & = \phi(A)
\end{align*}
\]  

(2.15)

where \( a_j \) is the activation value, \( b_j \) is the output of \( j^{th} \) hidden neuron and \( A \) is the output before the last activation function.

In the case of a regression problem, we want to update the weights through backpropagation according to the chosen loss function, in this case the RSS:

\[
E = \sum_{n=1}^{N} (y_n - \hat{y}_n)^2 = \sum_{n=1}^{N} (y_n - \phi(A_n))^2
\]  

(2.16)
We compute \( \frac{\partial E}{\partial W_j} \) as follows:

\[
\frac{\partial E}{\partial W_j} = \sum_{n=1}^N 2(y_n - \phi(A_n)) \cdot \frac{\partial (y_n - \phi(A_n))}{\partial W_j} \\
= \sum_{n=1}^N 2(y_n - \phi(A_n)) \cdot (-\phi'(A_n)) \cdot \frac{\partial A_n}{\partial W_j} \\
= \sum_{n=1}^N 2(y_n - \phi(A_n)) \cdot (-\phi'(A_n)) \cdot b_{j,n} \tag{2.17}
\]

and we obtain the backpropagation update rule for the \( j^{th} \) \( W_j \):

\[
W_{j}^{k+1} = W_{j}^{k} + 2\eta \sum_{n=1}^N (y_n - \hat{y}_n) \cdot \hat{y}_n' \cdot \rho (\sum_{i=1}^I w_{ji} \cdot x_{i,n}) \tag{2.18}
\]

Figure 2.7: Illustration of a Feedforward Neural Network with \( I \) input features, \( J \) hidden units and a single output.
In the same way, we compute \( \frac{\partial E}{\partial w_{ji}} \) as follows:

\[
\frac{\partial E}{\partial w_{ji}} = \sum_{n=1}^{N} 2(y_n - \phi(A_n)) \cdot \frac{\partial(y_n - \phi(A_n))}{\partial w_{ji}} = \sum_{n=1}^{N} 2(y_n - \phi(A_n)) \cdot (-\phi'(A_n)) \cdot \frac{\partial A_n}{\partial w_{ji}} \\
= \sum_{n=1}^{N} 2(y_n - \phi(A_n)) \cdot (-\phi'(A_n)) \cdot W_j \cdot \frac{\partial b_{j,n}}{\partial w_{ji}} \\
= \sum_{n=1}^{N} 2(y_n - \phi(A_n)) \cdot (-\phi'(A_n)) \cdot W_j \cdot \rho'(a_{j,n}) \cdot \frac{\partial a_{j,n}}{\partial w_{ji}} \\
= \sum_{n=1}^{N} 2(y_n - \phi(A_n)) \cdot (-\phi'(A_n)) \cdot W_j \cdot \rho'(a_{j,n}) \cdot x_{i,n}
\]

(2.19)

and we obtain the backpropagation update rule for the \( j^{th}, i^{th} \) \( w_{ji} \):

\[
w_{ji}^{k+1} = w_{ji}^k + 2\eta \sum_{n=1}^{N} (y_n - \hat{y}_n) \cdot \hat{y}_n' \cdot W_j \cdot \rho' \left( \sum_{i=1}^{I} w_{ji} \cdot x_{i,n} \right) \cdot x_{i,n}
\]

(2.20)

In the case of a binary classification problem, we want to update the weights through backpropagation according to the chosen loss function, in this case the binary crossentropy:

\[
E = -\sum_{n=1}^{N} (y_n \log(\hat{y}_n) + (1 - y_n) \log(1 - \hat{y}_n))
\]

(2.21)
We compute $\frac{\partial E}{\partial W_j}$ as follows:

$$
\frac{\partial E}{\partial W_j} = -\sum_{n=1}^{N} \left( \frac{y_n}{\phi(A_n)} \cdot \frac{\partial \phi(A_n)}{\partial W_j} + \frac{1 - y_n}{1 - \phi(A_n)} \cdot \frac{\partial (1 - \phi(A_n))}{\partial W_j} \right)
$$

$$
= -\sum_{n=1}^{N} \left( \frac{y_n}{\phi(A_n)} \cdot \frac{\partial \phi(A_n)}{\partial W_j} + \frac{1 - y_n}{1 - \phi(A_n)} \cdot \frac{\partial (1 - \phi(A_n))}{\partial W_j} \right) \cdot \phi'(A_n)
$$

$$
= -\sum_{n=1}^{N} \left( \frac{y_n}{\phi(A_n)} - \frac{1 - y_n}{1 - \phi(A_n)} \right) \cdot \phi'(A_n) \cdot b_{j,n}
$$

(2.22)

and we obtain the backpropagation update rule for the $j^{th}$ $W_j$:

$$
W_{j+1}^k = W_j^k + \eta \sum_{n=1}^{N} \left( \frac{y_n}{y_n} - \frac{1 - y_n}{1 - \gamma_n} \right) \cdot \gamma'_n \cdot \rho \left( \sum_{i=1}^{I} w_{ji} \cdot x_{i,n} \right) \tag{2.23}
$$

In the same way, we compute $\frac{\partial E}{\partial w_{ji}}$ as follows:

$$
\frac{\partial E}{\partial w_{ji}} = -\sum_{n=1}^{N} \left( \frac{y_n}{\phi(A_n)} \cdot \frac{\partial \phi(A_n)}{\partial w_{ji}} + \frac{1 - y_n}{1 - \phi(A_n)} \cdot \frac{\partial (1 - \phi(A_n))}{\partial w_{ji}} \right)
$$

$$
= -\sum_{n=1}^{N} \left( \frac{y_n}{\phi(A_n)} \cdot \frac{\partial \phi(A_n)}{\partial w_{ji}} + \frac{1 - y_n}{1 - \phi(A_n)} \cdot \frac{\partial (1 - \phi(A_n))}{\partial w_{ji}} \right) \cdot \phi'(A_n)
$$

$$
= -\sum_{n=1}^{N} \left( \frac{y_n}{\phi(A_n)} - \frac{1 - y_n}{1 - \phi(A_n)} \right) \cdot \phi'(A_n) \cdot W_j \cdot \frac{\partial b_{j,n}}{\partial w_{ji}}
$$

$$
= -\sum_{n=1}^{N} \left( \frac{y_n}{\phi(A_n)} - \frac{1 - y_n}{1 - \phi(A_n)} \right) \cdot \phi'(A_n) \cdot W_j \cdot \rho'(a_{j,n}) \cdot \frac{\partial a_{j,n}}{\partial w_{ji}}
$$

$$
= -\sum_{n=1}^{N} \left( \frac{y_n}{\phi(A_n)} - \frac{1 - y_n}{1 - \phi(A_n)} \right) \cdot \phi'(A_n) \cdot W_j \cdot \rho'(a_{j,n}) \cdot x_{i,n}
$$

(2.24)

and we obtain the backpropagation update rule for the $j^{th},i^{th}$ $w_{ji}$:

$$
w_{ji}^{k+1} = w_{ji}^k + \sum_{n=1}^{N} \left( \frac{y_n}{y_n} - \frac{1 - y_n}{1 - y_n} \right) \cdot \gamma'_n \cdot W_j \cdot \rho' \left( \sum_{i=1}^{I} w_{ji} \cdot x_{i,n} \right) \cdot x_{i,n} \tag{2.25}
$$
2.1. Deep Learning

Regardless of the task, putting all together we have obtained a powerful multilayer architecture able to learn complex, high-dimensional and non-linear mappings from a collection of observed samples, and this is true for a wide set of input data types.

2.1.5 Convolutional Neural Networks

In traditional Feedforward Neural Networks, the model extracts relevant features from the input and reduces the impact over the output of irrelevant ones, by connecting each neuron of each layer to each neuron of the next layer. For this reason, they are also named Fully-Connected layers or Dense layers.

While dealing with images, this input is characterized by a single matrix of numbers in case of greyscale images, or by a 3D tensor for RGB images, i.e. one matrix as before, but one for each colour channel. Feeding directly raw data in the such a network, especially in case of images with huge dimensions, leads to an enormous amount of weights to compute, with a consequent computational inefficiency: for instance, even if we consider a set composed by 100x100 images and a network with a hidden layer composed by 100 artificial neurons, this will already lead to 1 million of weights to compute. Moreover, there are data structures with features for which order is fundamental, e.g. timestamp sequentiality in time series or pixel position in images, but Feedforward Neural Networks are unable to discriminate the input topology and behave accordingly.

Due to these limitations, LeCun et al. (1995) designed a new network architecture based on layers named Convolutional layers and Pooling layers able to locally extract and combine features, making the computation feasible again. This model is called Convolutional Neural Network (CNN) and it is an architecture capable of filtering input data, layer after layer, and thus extracting various useful information to be exploited during the prediction phase. For simplicity, we will deeply describe only the mechanism behind a 2D-CNN, i.e. a locally connected network able to process images, but the same technique can be applied also over 1D- and 3D-CNNs.

The main ingredient of a 2D-CNN is the Convolutional layer. It is composed by a set of $K$ learnable filters $F$, also called kernels, which are matrices of numbers used to compute the 2D convolution function over a set of input images $I$. A 2D convolution is a function that, given
a generic $W \times H$ grey scale image $I$ and the $k^{th}$ filter $f^k$, iteratively produces single pixels $C^k_{m,n}$ belonging to the output image $C^k$ computed as:

$$C^k_{m,n} = (I \ast f^k)_{m,n} = \sum_{d=1}^{D} \sum_{e=1}^{E} f^k_{m,n}I_{m-d,n-e} \quad (2.26)$$

where $m = 1...M$ and $n = 1...N$ are the row and column indices of $C^k$, $D$ and $E$ are kernel dimensions and $\ast$ is the convolution operation.

As shown in Figure 2.8, the kernel shifts over the input image moving one pixel at a time for each iteration step. We call this number of pixels stride and indicate it with $S$. Considering also a practice called padding, which consists of framing the matrix representing the image inside a frame of zeroes of thickness $P$, we can compute the output dimensions as:

$$M = \left\lfloor \frac{H + 2P - D}{S} + 1 \right\rfloor$$
$$N = \left\lfloor \frac{W + 2P - E}{S} + 1 \right\rfloor \quad (2.27)$$

In the same way as Dense layers, also Convolutional layers update their weights according to the gradient descent technique.

Moreover, it is proven that the use of non-linear activation functions improves Convolutional layers performance. Thanks to the works of Hahnloser et al. (2000), Jarrett et al. (2009) and Nair and Hinton (2010) the Rectified Linear Unit (ReLU) is nowadays the most used
2.1. Deep Learning

23

activation function used in CNNs, and is defined as follows:

\[
ReLU(x) = \max(0, x)
\] (2.28)

The convolutional operations just presented are repeated for each filter belonging to the Convolution layer obtaining tensor \( C = \{C^1, \ldots, C^K\} \) with dimensions \( M \times N \times K \). The stacking of multiple Convolutional layers leads to the extraction of a wide range of features which, in many cases, may result redundant and lack of meaningful information. For this reason, and to further lighten the computational load within the CNNs, the Convolutional layers alternate with the so-called Pooling layers.

Similar to the Convolutional layers, the Pooling layers are responsible for reducing the spatial size of the convolved features through dimensionality reduction. There are two main types of Pooling: Max Pooling and Average Pooling. Max Pooling is the most used, and returns the maximum value from the portion of the image covered by a filter, as shown in Figure 2.9. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the kernel.

Given an input tensor \( T \) with dimensions \( M \times N \times K \) and a 2D-Pooling function \( P \) with a kernel \( p \times p \) and stride \( S = p \), the output tensor \( P(T) \) has dimensions \( \lceil \frac{M}{p} \rceil \times \lceil \frac{N}{p} \rceil \times K \). Through this mechanism, Pooling layers also perform noise suppression. In the same way as Dense and Convolutional layers, also Pooling layers update their weights according to the gradient descent technique. The backpropa-
gation algorithms for networks containing Convolutional and Pooling layers are deeply explained by LeCun et al. (1995) in his work, and represent an adaptation of the previously explained backpropagation algorithm for Feedforward Neural Networks.

2.1.6 Recurrent Neural Networks

Convolutional Neural Networks have demonstrated their ability to learn hierarchical feature representations in many contexts and are currently the first choice for many computer vision tasks such as Image Recognition or Video Analysis. This is possible, as we have seen, thanks to the matrix structure of the input data that can be efficiently exploited by these networks through a series of layers of learnable filters. It often happens that we have to deal with data that share a form of sequencing that must be taken into consideration. For this type of data, there are networks called Recurrent Neural Networks (RNN) able to exploit this type of information and learn temporal patterns.

RNNs are an extension of Feedforward Neural Networks, augmented with the ability to pass information through recurrent edges that span adjacent samples, introducing a notion of sequentiality to the model. The structure of an RNN is very similar to that of the standard Multilayer Perceptron, where we add feedback connections to the hidden units. Through these connections the model can retain information about the past, enabling it to discover correlations between events that are far away from each other in the data.

According to Elman (1991) notation, we formalize the simplest RNN composed by an input layer, an hidden recurrent layer and an output layer, and its general formulation is the following equation:

\[
\begin{align*}
    h_t &= \sigma_h(W_h x_t + U_h h_{t-1} + b_h) \\
    y_t &= \sigma_y(W_y h_t + b_y)
\end{align*}
\]  

(2.29)

where \( h_t \) is the output of the hidden recurrent layer at step \( t \), \( \sigma_h \) is the hidden activation function, \( W_h \) is the weight matrix of the hidden layer associated at the input \( x_t \) at step \( t \), \( U_h \) is the recurrent weight matrix of the hidden layer associated at the context state \( h_{t-1} \) at step \( t - 1 \) and \( b_h \) is the hidden bias, while \( \sigma_y \) is the output activation function, \( W_y \) is the weight matrix of the output layer, \( b_y \) is the output bias and \( y_t \) is the output of the RNN at step \( t \).
Feedforward Neural Networks are universal function approximators but RNNs are way more powerful. As demonstrated by Siegelmann and Sontag (1991), a finite sized RNN is able to simulate a universal Turing machine if the hidden activations are sigmoidal functions $\sigma$ defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$  

Specifically, any function computable by a Turing machine can be computed by a RNN with a finite size, in other words it is Turing-complete.

The most effective way to visualize the dynamics of this type of architectures across time steps is to unfold the network as shown in Figure 2.10. Exploiting its similarities with the Feedforward Neural Network, this unfolded architecture can be trained across the time steps using the backpropagation algorithm known with the name of backpropagation through time, designed by Werbos (1990). Unfolding RNNs takes advantage of the concept of parameter sharing, in fact, the weights are shared across different instances of the artificial neurons, each associated with different time steps. This allows to generalize to sequences of different lengths because the same weights are re-used at each time step. The core idea is that it is not important the absolute time step at which an event occurs, but it only makes sense to consider the event in some context that somehow captures what happened before.

Despite all these aspects that make the model just described very powerful, there are problems related to the learning phase. In fact, when an RNN has to manage long-term dependencies it tends to be very inaccurate due to two phenomena known as vanishing gradient and exploding gradient. Due to the application of the chain rule, the gradient of the error function is given by the multiplication of several terms and this can cause the gradient to rapidly vanish to zero or even to explode making impossible to effectively train the network. Normal-
izing input data and using hidden activation functions such as sigmoid or hyperbolic tangent functions, with their derivatives in range (-1,1) and (0,1) respectively, it is possible to face the problem of the exploding gradient. Computing the gradients with respect to the weights of the \( k \)th layer consists of multiplying \( k \) derivatives of values less than 1, meaning that the gradient will decrease exponentially with \( k \).

To overcome the problem of the vanishing gradient, Hochreiter and Schmidhuber (1997) introduced the Long-Short Term Memory architecture. This model resembles a standard neural network with a recurrent hidden layer, in which each node in the hidden layer is replaced with a memory cell. The LSTM architecture consists of a memory cell that can maintain its state over time, and some non-linear gating units which regulate the information flow into and out of the cell. The key feature of LSTM nodes is their ability to add or remove information to the cell state by means of these simple structures called gates.

With reference to Equation (2.29) and to Figure 2.11, it is possible to define the behaviour of this model as follows:

\[
\begin{align*}
\text{Input gate:} & \quad i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \\
\text{Forget gate:} & \quad f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \\
\text{Input node:} & \quad \tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \\
\text{Internal state:} & \quad c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
\text{Output node:} & \quad o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \\
\text{Output gate:} & \quad h_t = o_t \odot \tanh(c_t)
\end{align*}
\]
2.1. Deep Learning

Where ⊙ is the element-wise multiplication, σ is the sigmoid function described in Equation (2.30), tanh is the hyperbolic tangent function defined as:

\[ \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \]  

(2.32)

With this architecture, the LSTM is able to keep trace of long-term sequences thanks to the internal state that is updated and forwarded at each time step \( t \). It is also able to remove unnecessary information from it thanks to the forget state, while its prediction is forwarded thanks to the output gate. All this allows Deep Neural Networks based on these architectures to excel in tasks such as Speech Synthesis and Recognition, Time-Series Prediction, Natural Language Processing and Sentiment Analysis.
2.2 Camera Geometry

Camera Geometry represents that set of mathematical and algorithmic models designed to relate the coordinates of a point in a three-dimensional world with its projection on a two-dimensional plane. These models represent the starting point from which all the more complex models that will be presented in the rest of the thesis originated; it is, therefore, our intention to give a brief but detailed explanation.

Figure 2.12: Basic model of a single camera projecting a point belonging to the three-dimensional world to a two-dimensional plane (From Klette et al. (1995)).

In this section, we will present the basic background regarding Image Formulation, Epipolar Geometry and Multi-View Geometry. First, we will see how an image projects on a standard camera, by describing the classical Pinhole camera model and the transformations involved in the algorithm. Then we will extend the idea to two- and multiple-view cases, where more than one camera looks at the same scene. By this way, we will finally understand why and how multiple cameras are necessary to correctly reconstruct a 3D structure of a general scene.
2.2. Camera Geometry

2.2.1 Image Formation

One of the essential requirements in most of Computer Vision applications is understanding how images have been generated. For instance, in order to reconstruct the 3D geometry of a scene, we necessarily need a model that describes how the scene is projected on the images. The classical approach to describe Image Formation relies on the so-called Pinhole camera model. This model is a very effective compromise between mathematical simplicity and expressiveness.

We can assume that the world reference frame has its center \( C \) in a point called the camera center. From this point we generate three orthogonal axes, two of them called \( X \) and \( Y \) oriented in order to be parallel to the image plane, and the third, that we call \( Z \) oriented along the principal axis of the camera, as shown in Figure 2.13. We finally define as focal length \( f \) the distance between the camera center and the image plane. With basic geometry, the relation between point \( X = \{X, Y, Z\} \) and \( x \) can be expressed as follows:

\[
x = \begin{pmatrix} fX/Z \\ fY/Z \\ f \end{pmatrix}
\]

(2.33)

The \( x \) and \( y \) coordinates of this point represent the position of \( x \) in the image plane, where the 2D reference frame is defined by the axis \( x_{\text{cam}} \) and \( y_{\text{cam}} \), which lay on the image plane and intersect the principal axis. By representing \( X \) and \( x \) in homogeneous coordinates, the Equation (2.33) becomes:

\[
x_{\text{cam}} = \begin{pmatrix} x_{\text{cam}} \\ y_{\text{cam}} \\ 1 \end{pmatrix} = \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}
\]

(2.34)

Since an image is represented with a matrix, coordinates \((0, 0)\) can’t be considered as its center and therefore a more practical reference frame on the image plane has to be defined.
By calling $p$ the image plane center with coordinates $p_x$ and $p_y$, as shown in Figure 2.13, the Equation (2.34) becomes as follows:

$$
\mathbf{x}_{\text{image}} = \begin{pmatrix} x_{\text{cam}} + p_x \\ y_{\text{cam}} + p_y \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}
$$

(2.35)

At this point, we can define the matrix $\mathbf{K}$ as follows:

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

(2.36)

We call it *intrinsic camera* matrix, and $f$, $p_x$ and $p_y$ are its *intrinsic parameters*.

The idea of a world reference frame coinciding with the camera frame is an assumption of a very particular case. We can generalize the previous relationship between a 3D point in the generic world reference frame $W$ and the 2D point in the image plane where the 3D point is projected. Let $\mathbf{C}^W$ be the center of the camera in the world reference $W$ and $\mathbf{R}^W_C$ the rotation of the camera frame $C$ with respect to $W$. 
Then $E_{cw}^W$ is the matrix that describes the roto-translation of the camera with respect to the world reference frame, and is expressed as follows:

$$E_{cw}^W = \begin{pmatrix} R_{cw}^W & C_{cw}^W \\ 0 & 1 \end{pmatrix}$$

(2.37)

The 3D point in Equation (2.35) is in the camera reference frame, but usually 3D points need to be expressed in world coordinates. Therefore we rewrite the equation:

$$\mathbf{x}_{image} = (K \ 0) \mathbf{X}_c^c = (K \ 0) \ (E_{cw}^W)^{-1} \mathbf{X}_w^w$$

(2.38)

We define the inverse matrix of $E_{cw}^W$ as follows:

$$E = (E_{cw}^W)^{-1} = E_{cw}^c = \begin{pmatrix} R_{cw}^c & -R_{cw}^c C_{cw}^c \\ 0 & 1 \end{pmatrix}$$

(2.39)

$E$ is usually referred as **extrinsic camera** matrix: it is a roto-translation that translates points in world coordinates to the camera reference frame. Finally, the projection of a 3D point belonging to the world to the image plane is computed as:

$$\mathbf{x}_{image} = (K \ 0) \mathbf{E} \mathbf{X}_w^w = \mathbf{P} \mathbf{X}_w^w$$

(2.40)

$P$ is the **calibration matrix** which expresses both the intrinsic parameters of a specific camera (intrinsic matrix) and its the position and orientation (extrinsic matrix).

### 2.2.2 Epipolar Geometry

If in a reconstruction process are involved two views of the same scene we are talking about **epipolar geometry**; the Pinhole camera model described so far has to be updated in order to include the information gained from the presence of a second camera, and needs to remain independent from the structure of the observed scene and to rely only on cameras parameters.

Let $C_1$ and $C_2$ be two cameras, defined by the intrinsic matrices $K_1$ and $K_2$ and the extrinsic matrices $E_1$ and $E_2$; let $O_1$, $O_2$ be their optical centers and $r_1$, $r_2$ be their visibility rays, respectively. A point $X$, visible from both the cameras, is projected in the 2D points $x_1$ and $x_2$ belonging to $C_1$ and $C_2$ image planes, respectively. Then, let
assume camera centers $O_1$ and $O_2$, the points $x_1$ and $x_2$ and $X$ lay on the same plane $\pi$ (Figure 2.14) named **epipolar plane**. If the information about the point $X$ is represented only by the projection $x_1$, then the projection on $C_2$ image plane of the corresponding 3D point is assumed to be constrained to lay on the line $l_2$, named **epipolar line**, which is the projection of the viewing ray from $C_1$ to $x_1$, and vice versa is $X$ is represented only by the projection $x_2$.

The epipolar geometry of two views is synthesized into the so-called **fundamental matrix**, which represents a projective transformation that maps the point belonging to the image plane of one camera, to the lines of the other camera. Given two cameras with calibration matrices $P_1$ and $P_2$, we look for the mapping between a point $x_1$ located in the image plane of the first camera and the corresponding epipolar line $l_2$. The line $l_2$ is the projection on the image plane of the second camera of the line through the center $O_1$ and the point $x_1$ in world coordinates, $P_1^+x_1$, where $P_1^+$ is the pseudo-inverse of $P_1$ such that $P_1P_1^+ = I$. Therefore, $l_2$ pass through the projection of these two points, $P_2O_1$ and $P_2P_1^+x_1$, and is defined as:

$$l_2 = P_2O_1 \times P_2P_1^+x_1 \quad (2.41)$$

We define the epipole as $e_2 = P_2O_1$. It is now possible to derive the fundamental matrix $F$ as:

![Figure 2.14: Scheme of epipolar geometry model](image)
2.2. Camera Geometry

\[ l_2 = e_2 \times P_2 P_1^+ x_1 = [e_2] \times (P_2 P_1^+) x_1 = F x_1 \]  \hspace{1cm} (2.42)

If instead we know the mapping between 2D points on the two image planes, we know that \( x_1 \) and \( x_2 \) are the projection of the same point, the following relation, named epipolar constraint is valid:

\[ x_2^T F x_1 = 0 \]  \hspace{1cm} (2.43)

Indeed the point \( x_1 \) induces the epipolar line \( l_2 = F x_1 \) and the point \( x_2 \) lays on this line, \( x_2^T l_2 = F x_1 = 0 \).

2.2.3 Triangulation

In an ideal scenario, the epipolar geometry of the two cameras is perfectly known and thus, the fundamental matrix \( F \) and the other parameters perfectly allow to compute \( X \) position, as explained before. In real cases the 2D projections are affected by noise, the visibility rays are skew and the intersection may not exist, and thus the 3D position of \( X \) must be estimated.

The simplest approach to estimate the 3D point \( X \) given the measurements \( x_1 \) and \( x_2 \) in two cameras whose calibration matrices are \( P_1 \) and \( P_2 \) is the so called linear triangulation. For the first camera, we can rewrite the previous relations as:

\[
\begin{cases}
  x_1 (p_1 3^T X) - (p_1 1^T X) = 0 \\
  y_1 (p_1 3^T X) - (p_1 2^T X) = 0 \\
  x_1 (p_1 2^T X) - y_1 (p_1 1^T X) = 0
\end{cases}
\]  \hspace{1cm} (2.44)

where \( P_1 = \begin{pmatrix} p_1^1 \\ p_1^2 \\ p_1^3 \end{pmatrix} \) and \( P_2 = \begin{pmatrix} p_2^1 \\ p_2^2 \\ p_2^3 \end{pmatrix} \). An analogous relation can be written for the second camera. The resulting system is the following:

\[
\begin{pmatrix}
  x_1 p_1 3^T - p_1 1^T \\
  y_1 p_1 3^T - p_1 2^T \\
  x_1 p_1 2^T - y_1 p_1 1^T \\
  x_2 p_2 3^T - p_2 1^T \\
  y_2 p_2 3^T - p_2 2^T \\
  x_2 p_2 2^T - y_2 p_2 1^T
\end{pmatrix} X = 0
\]  \hspace{1cm} (2.45)

is linear and can be solved by SVD decomposition.
The previous method results in the minimization of an algebraic error, while a more accurate estimation can be achieved by minimizing the geometric error, i.e. by minimizing the error directly on the domain where lays the noisy measurements. Let $x_1$ and $x_2$ be two noisy measurements of the point $X$ in two cameras: the noise implies that they usually do not satisfy the epipolar constraint. In order to estimate $X$ we ideally look for the two points $\hat{x}_1$ and $\hat{x}_2$, projection of the estimate $\hat{X}$ that satisfy $\hat{x}_2^T F \hat{x}_1 = 0$ and minimize the function:

$$d(x_1, \hat{x}_1)^2 + d(x_2, \hat{x}_2)^2$$

(2.46)

where $d(x_1, x_j)$ is the Euclidean distance the points $x_1$ and $x_j$. To find the optimal solution that minimizes the reprojection error (2.46), we need to rewrite it conveniently as a function of a single variable. First, let notice that the constraint $\hat{x}_2^T F \hat{x}_1 = 0$ implies that the point $\hat{x}_2$ lies on the epipolar line $l_1$, and $\hat{x}_1$ lies on the epipolar line $l_2$. We can write (2.46) as:

$$d(x_1, l_1(t))^2 + d(x_2, l_2(t))^2$$

(2.47)

We parametrize line $l_1(t)$ by $t$ and we compute the corresponding epipolar line $l_2(t)$ through the fundamental matrix $F$. Equation (2.47) now becomes:

$$d(x_1, l_1(t))^2 + d(x_2, l_2(t))^2$$

(2.48)

which is a function of a single variable $t$. The minimization problem can now be solved by finding the root of polynomial function 2.48.

### 2.2.4 Multi-View Geometry

In case of multiple cameras that see the same point $X$, we want to take into account all the noisy measurements of the 3D point in order to achieve a more accurate and robust estimate of it. Equation (2.46) for $N$ cameras becomes:

$$\sum_{i=1}^{N} d(x_i, \hat{x}_i)^2$$

(2.49)

where $x_i$ is the measurements of the point $X$ in the $i$-th camera and $\hat{x}_i$ is the reprojection of $X$ on the same camera $i$. By making the projection explicit:
2.2. Camera Geometry

\[
C(X) = \sum_{i=1}^{N} d(x_i, P_iX)^2 = \sum_{i=1}^{N} ||x_i - proj_i(X)||^2 \tag{2.50}
\]

where \(proj_i(X) = P_iX\) is the projection function for the \(i\)-th camera. Therefore, the optimal method described in the previous paragraph can not be applied. Classical methods to optimize \(C\) and estimate the 3D position of \(X\) are the gradient descent, the Gauss-Newton and the Levenberg-Marquardt algorithms. They are iterative methods bootstrapping from an initial guess \(X_0\); at the \(k\)-th iteration the estimate of the parameter \(X\) can be written as:

\[
X_k = X_{k-1} + \Delta_{k-1} \tag{2.51}
\]

The three methods differs on how they compute the value of the update \(\Delta_{k-1}\). In the gradient descent, the update follows the decreasing direction of the function computed as the inverse of the gradient evaluated at the current value of \(X\):

\[
\Delta_{k-1} = -\gamma \nabla proj(X_{k-1}) \tag{2.52}
\]

where \(\gamma\) is a weighting coefficient. The Gauss-Newton method computes the decreasing direction by approximating the function with the Taylor expansion; the update becomes:

\[
\Delta_{k-1} = J_{C}^{-1} C(X_{k-1}) = -(J_{proj}^T J_{proj})^{-1} J_{proj}^T C(X_{k-1}) \tag{2.53}
\]

where \(J_{C} = \frac{\partial C}{\partial X}\) is the Jacobian of \(C\) and \(J_{proj} = \frac{\partial proj}{\partial X}\) is the Jacobian of the reprojection function \(proj\). Finally, the Levenberg-Marquardt method combines the two previous with a coefficient \(\lambda\), also known as damping factor:

\[
\Delta_{k-1} = -(J_{proj}^T J_{proj} - \lambda \text{diag}(J_{proj}^T J_{proj}))^{-1} J_{proj}^T C(X_{k-1}) \tag{2.54}
\]

The dumping factor is usually changed for each iteration in order to act similarly to gradient descent in the neighbourhood of the minimum, and similarly to Gauss-Newton when we are far from the minimum, as explained by Lourakis et al. (2005). The Jacobian \(J_{proj}\) is:

\[
J_{proj} = \frac{\partial proj}{\partial X} = \begin{pmatrix}
\frac{p_0^1 z - p_0^2 z}{z^2} & \frac{p_0^1 z - p_0^2 z}{z^2} & \frac{p_0^1 z - p_0^2 z}{z^2} \\
\frac{z^2}{z^2} & \frac{z^2}{z^2} & \frac{z^2}{z^2} \\
\frac{z^2}{z^2} & \frac{z^2}{z^2} & \frac{z^2}{z^2}
\end{pmatrix} \tag{2.55}
\]

with \((x, y, z)^T = P_iX\).
2.3 Sparse Reconstruction

Both Structure from Motion (SfM) in Computer Vision and Visual Simultaneous Localization and Mapping (V-SLAM) in Robotics face the problem of reconstructing a sparse map of the environment together with the position and orientation of the cameras that captured the scene. In the early stage, the two communities focused on different aspects. SfM aims at estimating accurately the map, also called structure, of a generic set of images with no particular time constraint, therefore it processed all the images at the same time grouping them in a batch. On the other hand, V-SLAM was thought to be deployed on a robot, therefore, the main goal was to obtain a fast and accurate estimate of the robot pose with respect to the environment by processing a video sequence. SLAM algorithms need to perform in real-time, i.e. at the same rate of the camera, and incrementally, i.e. a new image has to be included in the estimation as soon as it becomes available.

These different approaches to the same problem led the researchers of the two communities to develop deeply different algorithms which rely on different estimation tools. Classical Structure from Motion algorithms exposed by Triggs et al. (1999), Sibley et al. (2009) and Wu et al. (2011) extract a set of 2D points for each image associated with a descriptor, e.g. SIFT by Lowe (2004) or ORB by Rublee et al. (2011), then they find the correspondences of these 2D point among all the images being likely generated by the same 3D point. Bootstrapping from an initial guess, SfM globally optimizes the poses of the cameras and the 3D point positions, to minimize the error between the 2D points measurements and the projection of the current estimate of the 3D points on the cameras, i.e. the reprojection error $e_r$ shown in Figure 2.15. Dealing with $n$ cameras and $k$ 3D points, we express with $x^{2D}_{ij}$ the measurement of point $i$ in image $j$, and with $\Pi_j(x^{3D}_i)$ the projection of the 3D point $i$ on the image plane of the $j^{th}$ camera, SfM aims at minimizing the following error function:

$$\sum_{i=0}^{n} \sum_{j=0}^{k} ||x^{2D}_{ij} - \Pi_j(x^{3D}_i)||^2 = \sum_{i=0}^{n} \sum_{j=0}^{k} e_r(i,j)^2. \quad (2.56)$$

This function is usually minimized through the Gauss-Newton or the Levenberg-Marquardt algorithms, and the process is named as Bundle Adjustment, since the minimization process adjusts the bundles...
of camera-to-point viewing rays, by jointly moving the camera poses and 3D point positions. This complete process requires a significant computational effort that nowadays is made feasible thanks to GPU-processing.

The SLAM approaches proposed by Davison et al. (2007), Ceriani et al. (2014) and Grasa et al. (2011), adopt a different point of view, focusing on the efficiency of the estimate, and thus they use a different estimation tool, i.e. the Extended Kalman Filter (EKF) which estimates iteratively a state, composed by the camera pose with respect to the world reference frame together with the map of the environment, by incorporating the new measurement, i.e. the $x_{ij}^{2D}$, and marginalizing the old camera poses. With the Parallel Tracking and Mapping (PTAM) algorithm proposed by Klein and Murray (2007), these two approaches become closer. PTAM introduced bundle adjustment optimization in a SLAM system decoupling the fast-tracking process, i.e. the camera pose estimation, and the slow mapping processes in two parallel threads. The tracking thread processes each frame of the video sequence while the mapping thread works only on the so-called keyframes. This new paradigm, named keyframe-SLAM, has gained more and more interest in the robotics community, such that it became more popular with respect to the filtering approach presented by Strasdat et al. (2010). Moreover, the improvements on the bun-

Figure 2.15: Reprojection error $e_r$ of the 3D point $x_i^{3D}$ in camera $C_j$. 

2.3. Sparse Reconstruction
dle adjustment optimization process proposed by Kaess et al. (2008), the formalization of the SLAM problem as a factor graph proposed by Thrun and Montemerlo (2006), and the availability of optimization libraries such as g2o by Kümmerle et al. (2011) and GTSAM by Dellaert (2012) led the researchers to move from the EKF estimation to keyframe-SLAM based on the works of Strasdat et al. (2011), Sünderhauf and Protzel (2012) and Johannsson et al. (2013). In conclusion, both SfM and SLAM algorithms are able to estimate the pose of the cameras and a point cloud representing the map of the environment: the former handles a generic set of data while the latter deals with a video sequence. Structure from Motion evolved into Multi-View Stereo algorithms able to reconstruct 3D scenes in a very accurate and detailed manner, by processing all the images at the same time and exploiting the recent improvements in GPU-computing. We will widely discuss this family of techniques in the next chapter.
Chapter 3

State of the Art

Both as regards Super-Resolution and Multi-View Stereo algorithms, the world of Computer Vision has made great strides especially in recent years. Think, for example, of the latest generation of video games, or the world of self-driving cars, both areas that use, albeit separately, both of these technologies. All these improvements were made possible by the parallel calculation based on the GPU, which allowed the execution of very complex algorithms whose computation was unthinkable without the help of a Supercomputer even just 10 years ago.

In this chapter our goal is to explain what is meant by Single-Image Super-Resolution and Multi-View Stereo techniques, starting from the basic concepts to be combined with those proposed in the previous chapter, up to give a detailed idea of the models that will be used in our experiments.

More in detail, we will begin by describing both tasks from a general point of view, formalizing their purpose and illustrating their first historical approaches. Next, we will show the steps taken by the scientific community to optimize the performance of both families of algorithms by exploiting the better hardware performance of the last few years, up to the latest trends based on complex heuristic algorithms and the most updated architectures based on Artificial Neural Networks.

In this chapter, we will discuss what Single-Image Super-Resolution and Multi-View Stereo techniques are, giving an explanation about what kind of problem they solve, briefly walking through the paths that have lead to the birth of recent algorithms and, finally, deeply describing the best current pipelines, the state-of-the-art.
3.1 Single-Image Super-Resolution

Super-Resolution is a classical Computer Vision task with the goal of restoring high-resolution images from one or more low-resolution observations of the same scene. According to the number of input LR images, the task can be classified into Single-Image Super-Resolution (SISR) and Multi-Image Super-Resolution (MISR). Compared with MISR, SISR is much more popular because of its higher efficiency. Since a Super-Resolution image with high perceptual quality has more valuable details, it is starting to be widely used in many areas, such as in medical imaging, satellite imaging, security imaging and in video games. An example of its application is shown in Figure 3.1.

In this section we will describe what is meant by SISR, giving a formulation to this often ill-posed problem and explaining its general functioning. Subsequently, we will deepen the various techniques that the Computer Vision community has developed over the years with the aim of improving the temporal and perceptual performance of the results, starting from the first models proposed up to the most recent pipelines based on Deep Learning. Finally, we will compare the most advanced neural networks belonging to this category focusing on the model that we will use for our experiments in the next chapter.

![Figure 3.1: Single-Image Super-Resolution with scale factor 2 through the algorithm DBPN (by Haris et al. (2019)). On the left we have the cover of the Pulse album by Pink Floyd, on the right its super-resolution version.](image-url)
3.1. Single-Image Super-Resolution

3.1.1 Problem Formulation

Let $I_{LR}$ be a low-resolution image with width $w_{LR}$ and height $h_{LR}$, and $SR^k$ a generic Super-Resolution function with scale factor $k$, assumed being a positive integer greater than one. The SISR objective is to compute $I_{SR}^k$ such that:

$$I_{SR}^k = SR^k(I_{LR}) \approx I_{HR}^k$$

(3.1)

having that:

$$w_{SR}^k = kw_{LR} = w_{HR}^k$$
$$h_{SR}^k = kh_{LR} = h_{HR}^k$$

(3.2)

where $I_{HR}^k$ is the high-resolution version with scale factor $k$ of $I_{LR}^k$ image.

In order to quantify resolution improvements, we need to evaluate the result according to some metrics. For simplicity, we show the computations referred to a single gray-scale channel while, in case of RGB images, the following equations have to be applied on each channel.

Let $I_1$ and $I_2$ be two images with the same width $W$ and height $H$. The simplest metric is the classical Mean Squared Error (MSE):

$$MSE(I_1, I_2) = \frac{1}{WH} \sum_{w=1}^{W} \sum_{h=1}^{H} (I_1(w, h) - I_2(w, h))^2$$

(3.3)

It has the drawback to work only on images with values scaled in the same range, that in many cases may be very large reducing computation efficiency. It is widely preferred the usage of a derived metric, the Peak Signal Noise Ratio (PSNR), that can be considered a normalized version of MSE. Its formulation is the following:

$$PSNR(I_1, I_2) = 10 \log_{10} \left( \frac{R(I_1, I_2)}{MSE(I_1, I_2)} \right)$$

(3.4)

with

$$R(I_1, I_2) = \max(d(I_1), d(I_2))$$

(3.5)

where $d(I)$ is the maximum fluctuation in the input image $I$ data type. For instance, if the input image has a double-precision floating-point data type, then $d(I)$ is 1. If it has an 8-bit unsigned integer data type, $d(I)$ is 255, and so on. Another widely used metric to compare SISR
results is the *Structural Similarity* (SSIM). The difference with respect

The difference with respect to the previous ones is that these approaches estimate absolute errors, while SSIM is a perception-based metric that considers image degradation as perceived change in structural information, while also incorporating important perceptual phenomena, such as luminance masking and contrast masking terms.

Let $x$ and $y$ be two sliding windows with common size $L \times L$ moving on $\mathcal{I}_1$ and $\mathcal{I}_2$, respectively; let $\mu_x$ and $\mu_y$ be their means, $\sigma_x$ and $\sigma_y$ be their variances and $\sigma_{xy}$ be the covariance between $x$ and $y$. Let $D$ be the dynamic range of pixel-values, and $k_1 = 0.01$ and $k_2 = 0.03$ be two constant coefficients. Let's define two other coefficients $c_1 = (k_1 D)^2$ and $c_2 = (k_2 D)^2$ used to stabilize the metric in case of weak means and variances. Finally, we can state that:

$$SSIM(x, y) = \frac{(2\mu_x \mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

As explained in the beautiful work of Yang et al. (2019), the scientific community is mainly using these two last metrics to compare SISR performance, and thus we do the same.

Finally, the perceptual evaluation carried out by human eyes remains a qualitative metric to be taken into consideration also in light of the artifacts that these algorithms can produce. In this regard, it is possible to make a comparison of this type by observing the images in Figure 3.2. At this point, the evaluation criterion is fixed, but there is still a conceptual problem to solve.

In a real scenario it is not possible to know how much a $SR$ function has correctly approximated the high-resolution image $\mathcal{I}_{HR}$ because the latter simply does not exist. Moreover, it would have different dimensions with respect to $\mathcal{I}_{LR}$, and thus it would be not evaluable with previously described metrics. So instead of directly computing the goodness of a SISR algorithm, we need to estimate it.
3.1. Single-Image Super-Resolution

![Image](image)

(a) low-resolution (downsampling)  
(b) SR from low-resolution  
(c) high-resolution  
(d) SR from high-resolution

Figure 3.2: Single-Image Super-Resolution with scale factor 2 through the algorithm DBPN (by Haris et al. (2019)) applied over the cover of the Pulse album by Pink Floyd. The SISR assumption is that the improvement between (a) and (b) represents the best estimation of the improvement between (c) and (d).
Given an image $I$ with width $w$ and height $h$, let $F^{1/k}$ be a generic down-sampling function with scale factor $k$, that receives as input an image, maps it to a lower space and also degrades its information by adding some noise, so that:

$$I^{1/k}_F = F^{1/k}(I)$$

$$w^{1/k}_F = \frac{w}{k}$$

$$h^{1/k}_F = \frac{h}{k}$$

(3.7)

The best function in terms of efficiency and results quality, used in almost every SISR paper for this purpose, is the so-called bicubic interpolation.

With all these elements together, we can finally give a correct formulation of the SISR problem. Regardless of the used evaluation metric $\mathcal{M}$, given a low-resolution image $I_{LR}$, a Super-Resolution function $\mathcal{SR}$ and a bicubic interpolation function $\mathcal{BC}$ with a common scale factor $k$, we define the downgraded-resolution image $I_{DR}$ as explained in 3.7:

$$I^{1/k}_{DR} = \mathcal{BC}^{1/k}(I_{LR})$$

(3.8)

According to Equation (3.1), we compute the Super-Resolution of both the images:

$$I^k_{SR} = \mathcal{SR}^k(I_{LR}) \approx I^k_{HR}$$

(3.9)

$$\hat{I}_{LR} = \mathcal{SR}^k(I_{DR}^{1/k}) \approx I_{LR}$$

(3.10)

Finally, having both $\hat{I}_{LR}$ and $I_{LR}$, we can compute $\mathcal{M}(\hat{I}_{LR}, I_{LR})$ and assert that this is the best possible estimation of the $\mathcal{SR}$ algorithm metric on a real scenario:

$$\mathcal{M}(\hat{I}_{LR}, I_{LR}) = \hat{\mathcal{M}}(I^k_{SR}, I^k_{HR}) \approx \mathcal{M}(I^k_{SR}, I^k_{HR})$$

(3.11)

All the algorithms we show in next sections follow these evaluation assumptions.

### 3.1.2 Classical Super-Resolution

Depending on the technologies used to enhance and refine the high resolution details of the starting image, the Super-Resolution algorithms have been repeatedly divided into various categories. In accordance
with the classification made by Yang et al. (2019) SISR algorithms can be grouped in three macrocategories: interpolation-based methods, reconstruction-based methods and Learning-based methods.

Interpolation-based SISR algorithms are characterized by high performance in terms of speed and by simpler and lighter pipelines than other methods’ families. Some examples of algorithms go from the trivial linear and nearest-neighbour interpolations, to the more sophisticated Lanczos resampling by Duchon (1979) and the bicubic interpolation by Keys (1981). We will use the latter in our experiments, because it is the most used among the Computer Vision community due to its good trade-off between efficiency and accuracy, and thus we present an intuition of its mechanism. Let \( f(x, y) \) be a function and let \( f_x, f_y, f_{xy} \) be its partial derivatives which are known at the four corners of a square \( S \) with coordinates \( C = [(0, 0), (1, 0), (0, 1), (1, 1)] \). The surface to interpolate is described as:

\[
p(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j
\]

where the 16 coefficients \( a_{ij} \) are values to compute in order to solve the task. They can be found by solving, for each \( c \in C \) the equations \( f(c) = p(c), f_x(c) = p_x(c), f_y(c) = p_y(c) \) and \( f_{xy}(c) = p_{xy}(c) \). The so obtained bicubic-interpolated square \( S \) can be patched together with other bicubic surfaces belonging to an arbitrary sized rectangular grin, i.e. an image, obtaining a bicubic-interpolated image. It should be noted that despite the speed of calculation of this family of SISR algorithms, their main disadvantage lies in not achieving great performance.

Reconstruction-based SISR algorithms, such as the works made by Marquina and Osher (2008), Yan et al. (2015), Sun et al. (2008) and Dai et al. (2009), are generally based on mappings between low-resolution and Super-Resolution relying on strong prior knowledge to constraint output solution space. These family of methods are able to achieve better performance compared with interpolation-based ones, but have the drawbacks of being much more time consuming and of being not able to produce good results with high scale factors. Moreover, they lack on clarity in the definition of mapping functions between low-resolution and high-resolution spaces and turn out to be inefficient while trying to establish complex high dimensional mappings starting from data with huge dimensions.
The Learning-based SISR algorithms, also named example-based algorithms, rely on Artificial Intelligence techniques and are able to achieve the best performance in terms of time and quality among all the reconstruction-based methods, even when dealing with huge amount of data as well as with large input images. We divide them in DeepLearning-based (DL-based) and MachineLearning-based (ML-based) depending on whether they are based on Artificial Neural Networks, as in the first case, or not. ML-based SISR algorithms usually use statistical relationships to map low-resolution and high-resolution pairs learned during a training process. The first attempt is the Markov Random Field approach made by Freeman et al. (2002) that exploits for the first time the huge amount of available data for this purpose. Then, many other ML-based models have made their way in this area, such as the so called sparse coding methods, e.g. A+ by Timofte et al. (2014) or K-SVD by Aharon et al. (2006), or the Random Forest applied to SISR by Schulter et al. (2015), increasing the quality of the results from time to time and being able to reconstruct high-resolution images even with higher scale factors. In very last years, DL-based architectures are achieving even better results and nowadays they represent the state-of-the-art in this task.

3.1.3 DeepLearning-based Super-Resolution

Thanks to the recent improvements in GPU-processing, Artificial Neural Networks are now able to analyze incredibly huge amount of data, achieving a powerful generalization ability also in the SISR field thanks to their ability to learn non-linear hierarchical representation from input features.

The first successful attempt is the so called Super-Resolution Convolutional Neural Network (SRCNN) made by Dong et al. (2015). The architecture is a three-layer Convolutional Neural Network (CNN) for SISR that directly learns an end-to-end mapping between low-resolution and high-resolution image pairs, with a little pre- and post-processing beyond the optimization and with MSE as loss function to minimize. The authors have built an architecture in order to create a relationship between their model and traditional sparse-coding-based (SC) structure, giving to the layers the role of patch extractor, non-linear mapper and reconstructor, respectively. The result leads to a performance improvement in terms of PSNR, as shown in Figure 3.3,
3.1. Single-Image Super-Resolution

and also in terms of qualitative perception, as shown in Figure 3.4.

This architecture opens the doors to many ideas, also thanks to its
drawbacks: indeed, apart from being a totally new architecture in this
field, it is trained over bicubic-downsampled images instead that over
real low- and high-resolution images pairs, it minimizes a loss function
that is considered too much general for this specific task, and it is just
a three-layer architecture.

Starting from this idea, and thanks to the implementation of the
so called Deconvolutional layers by Zeiler et al. (2011), the FSRCNN
model is designed by Dong et al. (2016). The main idea behind this
network is to use these new layers to upsample data at the end of the
network, to speed up the computation by avoiding to carry on huge
tensors through the network. Indeed, Deconvolutional layers use an
arbitrary interpolation operator, e.g. the nearest neighbor, to project
the input to a larger hypothesis space, and then use the convolution
operator with stride equal to 1. Its main drawback consists in the
redundancy of these upsampled pixels, and for this reason Shi et al.
(2016) designed the Subpixel Convolutional layer known as ESPCN.
Rather than increasing resolution by explicitly enlarging feature maps,
this layer expands the channels of the output features for storing the
extra points to increase resolution and then rearranges these points to
obtain the high-resolution output through a specific mapping criterion.

Figure 3.3: SRCNN surpasses the PSNR score ($SR^2$) of both bicubic interpolation
and SC on the Set5 test set. (From Dong et al. (2015))
In last years, many DL-based SISR architectures have demonstrated a great effectiveness in exploiting the power of very deep networks, despite training difficulties. It is the case of the VDSR model designed by Kim et al. (2016) and based on the first 20-layers of the famous VGG-net by Simonyan and Zisserman (2014). The innovative contribution relies in the fact that this model is able to be used for multiple scale factors since SISR processes with different scale factors have demonstrated strong relationships between each other. Moreover, it takes bicubic downsampled input like SRCNN, but during the training VSDR learns from low-resolution inputs with different downsampling scale factors. This idea is replicated over different deep networks architectures, e.g. SRResNet by Ledig et al. (2017) inspired by the famous ResNet He et al. (2016) based on residual learning patterns to take advantage from inner layers outputs.

Among this huge amount of models, one of the most relevant and performing is surely EDSR made by Lim et al. (2017) that, starting
3.1. Single-Image Super-Resolution

Table 3.1: Comparison between several DL-based SISR algorithms over PSNR, SSIM, datasets used for training the models, number of parameters and number of multiplications and additions during training phase (From Yang et al. (2019))

<table>
<thead>
<tr>
<th>Models</th>
<th>PSNR/SSIM (×4)</th>
<th>Train data</th>
<th>Parameters</th>
<th>Mult &amp; Adds</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRCNN</td>
<td>30.49/0.8628</td>
<td>ImageNet subset</td>
<td>57k</td>
<td>52.5G</td>
</tr>
<tr>
<td>ESPCN</td>
<td>30.90/-</td>
<td>ImageNet subset</td>
<td>20k</td>
<td>1.43G</td>
</tr>
<tr>
<td>VDSR</td>
<td>31.35/0.8838</td>
<td>G200 + Yang91</td>
<td>665k</td>
<td>612.6G</td>
</tr>
<tr>
<td>SRResNet</td>
<td>32.05/0.9019</td>
<td>ImageNet subset</td>
<td>1.5M</td>
<td>127.8G</td>
</tr>
<tr>
<td>EDSR</td>
<td>32.62/0.8984</td>
<td>DIV2K</td>
<td>43M</td>
<td>2890.0G</td>
</tr>
<tr>
<td>LapSRN</td>
<td>31.54/0.8855</td>
<td>G200 + Yang91</td>
<td>812k</td>
<td>29.9G</td>
</tr>
<tr>
<td>DBPN</td>
<td>32.47/0.8980</td>
<td>DIV2K + Flickr + ImageNet subset</td>
<td>10M</td>
<td>5715.4G</td>
</tr>
</tbody>
</table>

from SRResNet has improved both time and metric performance by removing the batch normalization from residual units, by increasing the number of output features of each layer and by implementing the transfer learning while dealing with huge scale factors. Another noteworthy model, with a completely different nature from those mentioned so far, is ZSSR designed by Shocher et al. (2018). This architecture has the distinction of being the first DL-based algorithm for SISR to have internal-example training. In particular, no other images are required other than the one to predict, as this is processed to produce many low-resolution and high-resolution pairs of the same. The only flaw of this architecture is that it is not suitable for high scale factors due to the computation.

To achieve good performance also while restoring high-resolution images with very huge scale factors, e.g. 8x, many other ideas have been proposed such as methodologies based on incremental improvements, with the aim of gradually producing higher outputs that are used as input for successive steps, e.g LapSRN by Lai et al. (2017) that uses the Laplacian pyramid structure to reconstruct its high-resolution outputs. One of the last trends is based on deep architectures with an iterative back-projection system to update reconstruction error by feeding it back in the network to tune high-resolution results. The most important model belonging to this family is DBPN proposed by Haris et al. (2018), and we will widely describe it in next subsection.

Finally, there are also other trends that aim to improve the perceptual qualities at the expense of performance based on absolute error.
The most promising approaches are based on GANs, such as SRGAN designed by Ledig et al. (2017) or the more recent SFTGAN designed by Wang et al. (2018) but this family of models has not found yet a proper application field in Computer Vision. Indeed, despite their outputs with a very high perceptual quality, they may create unwanted artifacts for a correct and performing 3D reconstruction due to poor MSE and PSNR performance compared with the other methods. According to the great comparison made by Yang et al. (2019) and reported in Table 3.1, we choose the DBPN for our experiments because it is based on the most recent architectures and has several updated versions that achieves very high performance.

3.1.4 Deep Back-Projection Network

The Deep Back-Projection Network (DBPN) model designed by Haris et al. (2018) is one of the most performing SISR methods actually available in Computer Vision community. Thanks to its iterative back-projection mechanism, it is and end-to-end architecture able to iteratively compute the reconstruction error, to then use it to refine high-resolution images achieving excellent performance also while dealing with large scale factors, as shown in Figure 3.5.

![Figure 3.5: SISR result over SR8. PSNR of DL-based models: LapSRN (15.25 dB), EDSR (15.33 dB) and DBPN (16.63 dB) (From Haris et al. (2019))](image)
### 3.1. Single-Image Super-Resolution

![Architecture of basic Deep Back-Projection Network](image)

Figure 3.6: Architecture of basic Deep Back-Projection Network (From Haris et al. (2019))

The basic model architecture is composed by multiple iterative up- and down-sampling layers grouped into units and used to provide a projection error feedback mechanism, leading to numerous degraded and high-resolution hypothesis images that the network uses to improve the output results. Given an image pair $I_{LR}$ and $I_{HR}$ let $w_{LR} \times h_{LR}$ and $w_{HR} \times h_{HR}$ be their dimensions, respectively, so that $w_{LR} < w_{HR}$ and $h_{LR} < h_{HR}$. The so called projection unit is the core block of this architecture, and is trained to map low-resolution features to high-resolution ones in up-projection case, and vice versa in down-projection one. According to authors’ notation, the up-projection unit is defined as follows:

\[
\begin{align*}
\text{scale up:} & \quad H^t_0 = (L^{t-1} \ast p_t) \uparrow_s \\
\text{scale down:} & \quad L^t_0 = (H^t_0 \ast g_t) \downarrow_s \\
\text{residual:} & \quad e^t = L^t_0 - L^{t-1} \\
\text{scale residual up:} & \quad H^t_1 = (e^t \ast q_t) \uparrow_s \\
\text{output feature map:} & \quad H^t = H^t_0 + H^t_1
\end{align*}
\] (3.13)

and at the same way, down-projection unit is defined like this:

\[
\begin{align*}
\text{scale down:} & \quad L^t_0 = (H^t_0 \ast g^t_t) \downarrow_s \\
\text{scale up:} & \quad H^t_0 = (L^t_0 \ast p^t_t) \uparrow_s \\
\text{residual:} & \quad e^t = H^t_0 - H^t \\
\text{scale residual down:} & \quad H^t_1 = (e^t \ast g^t_t) \uparrow_s \\
\text{output feature map:} & \quad L^t = L^t_0 + L^t_1
\end{align*}
\] (3.14)
where $\ast$ is the convolution operator, $\uparrow_s$ and $\downarrow_s$ are respectively the up- and down-sampling operators with scale factor $s$, while $p_t, g_t, q_t$ and $p'_t, g'_t, q'_t$ are the (de)convolutional layers at stage $t$.

The entire basic model can be divided, as shown in Figure 3.6 in initial feature extraction, back-projection stages and reconstruction. Moreover, in the last model update, Haris et al. (2019) have adapted the model described since now to last Deep Learning trends by including dense projection units (DBPN-D), by treating single (DBPN-R) and multiple (DBPN-MR) projection units as recurrent layers and by implementing residual learning (DBPN-RES), obtaining improvements from each of these updates over the basic model.
3.2 Multi-View Stereo

Multi-View Stereo (MVS) is a Computer Vision field with the aim of reconstructing a 3D scene starting from a set of images for which it is assumed to know its camera poses, usually estimated by a Structure from Motion (SfM) or Simultaneous Localization and Mapping (SLAM) algorithm. Its main purpose is to recover a detailed dense and accurate reconstruction: while SfM and SLAM algorithms produce a sparse point cloud as a representation of the environment map, MVS aims at reconstructing the correct position of all the pixels of images, resulting in a dense or continuous environment model.

In this section, we will describe how this family of algorithms was born and on which hypotheses its most modest methods are based. After that, we will give an idea on how to calculate a metric to evaluate and subsequently optimize a 3D reconstruction and we will classify the various families of MVS algorithms. Next, we will explain what is meant by depth map and we will illustrate the functioning of the main families of techniques that are based on them in the 3D reconstruction phase. Finally, we will deepen the PatchMatch-based and DeepLearning-based pipelines and introduce the algorithms that we will use in our experiments in the next chapter.

Figure 3.7: 3D dense reconstruction of the Louvre museum made through the use of the PatchMatch-based Multi-View Stereo algorithm by COLMAP (from Schönberger et al. (2016))
3.2.1 Image Appearance

To understand how to compare different images to perform 3D reconstruction, we need to describe how each pixel of a single image is generated. The pixel intensity is proportional to the number of photons perceived by the camera sensor at that point, i.e. the amount of light that reaches it. This quantity, named irradiance, is proportional to the radiance of the scene which is the amount of light emitted by the objects. For a surface patch around a point $x$, the radiance in direction $v$ is defined by the so-called plenoptic function $L(x, v)$. The images are therefore a set of pixels whose values are observations of the plenoptic function. Many factors influence the value of a pixel: the overall scene illumination, the reflectance of the materials and the geometry of the objects perceived or how the camera sensor captures the light.

The estimation of illumination of a scene is a complex task, and thus in Computer Vision, each point is assumed as a light source. The reflectance is strictly related to the material of the surface, in fact, if its value does not depend on time and wavelength, and the material is exempt from scattering effects, then the constant of proportionality between the reflected radiance and the irradiance of a surface can be easily computed through the bidirectional reflectance distribution function (BRDF) defined by Cohen and Wallace (2012). In a non-ideal scenario, these conditions are not always met and therefore different BRDF models have been proposed to generalize the reflection based on the nature of the different materials of the scene.

Within most of the MVS models used in Computer Vision, the hypothesis of the Lambertian model, which is based on the idea of the perfect diffuser, is the most used. This is, in fact, able to effectively describe smooth objects whose splendour is independent of the point of view. According to this model, if the light vector $l$ is reflected by a surface with normal $n$ and constant albedo $\rho$, then the intensity of the radiance $L$ is

$$L = \rho \cdot |n||l| \cos(\theta),$$

(3.15)

where $\theta$ is the angle between $n$ and $l$ as shown in Figure 3.8.

In the real world there are very few materials that can form a perfectly reflective surface, but there are many others close enough to this assumption that they can be considered Lambertians. If we assume a point-wise light source in a Lambertian world, then, a 3D point projects to pixels with the same intensity on cameras where it is not occluded.
3.2. Multi-View Stereo

Let $I$ and $J$ be two images, and $\Pi_i(x^{3D})$ and $\Pi_j(x^{3D})$ the projections of the 3D point $x^{3D}$ in the two images, then:

$$I(\Pi_i(x^{3D})) = J(\Pi_j(x^{3D})).$$ (3.16)

This relation is the so called constant illumination assumption and it is widely exploited in Computer Vision, not only in Multi-View Stereo algorithms, but also in optical flow and in any algorithm for which pairwise pixel comparison is needed. Indeed, this assumption represents the basis to define photo-consistency measures.

### 3.2.2 Photo-Consistency

Multi-View Stereo aims at recovering 3D information from a set of calibrated images. The coherence and accuracy measure to evaluate and optimize the recovered 3D data is called photo-consistency, which measures the similarity between two image regions. Different photo-consistency measures have been proposed and they all rely on the constant illumination assumption.

Given two images $I_1$ and $I_2$ and two pixels $p \in I$ and $q \in J$, the most trivial photo-consistency measure is the difference between the intensity values $I_1(p_0)$ and $I_2(q_0)$ but, taking into account only a single value, it lacks in robustness. Robust photo-consistency measures consider a set of neighboring pixels around $p_0$ and $q_0$, i.e. $p_k$ and $q_k$ where $1 \leq i \leq K$, that usually form a rectangular region.
The most used measures are the Sum of Squared Differences (SSD):

$$SSD(I_1(p_0), I_2(q_0)) = \sum_{k=0}^{K} (I_1(p_k) - I_2(q_k))^2$$  \hspace{1cm} (3.17)

the Sum of Absolute Differences (SAD):

$$SAD(I_1(p_0), I_2(q_0)) = \sum_{k=0}^{K} |I_1(p_k) - I_2(q_k)|$$  \hspace{1cm} (3.18)

and the Normalized Cross-Correlation (NCC):

$$NCC(I_1(p_0), I_2(q_0)) = \frac{v_{I_1,I_2}(p_0, q_0)}{\sqrt{v_{I_1}(p_0)v_{I_2}(q_0)}}$$  \hspace{1cm} (3.19)

where:

$$v_{I_1,I_2}(p_0, q_0) = \frac{1}{K} \sum_{k=0}^{K} I_1(p_k)I_2(q_k) - \mu_{I_1}(p_0)\mu_{I_2}(q_0)$$  \hspace{1cm} (3.20)

is the correlation between the two patches $p_0$ and $q_0$,

$$\mu_{I_1}(p_0) = \frac{1}{K} \sum_{k=0}^{K} I_1(p_k)$$

$$\mu_{I_2}(q_0) = \frac{1}{K} \sum_{k=0}^{K} I_2(q_k)$$  \hspace{1cm} (3.21)

are the mean values of the pixels considered in images $I_1$ and $I_2$, respectively,

$$v_{I_1}(p_0) = \frac{1}{K} \sum_{k=0}^{K} (I_1(p_k))^2 - \mu_{I_1}(p_0)^2$$  \hspace{1cm} (3.22)

$$v_{I_2}(q_0) = \frac{1}{K} \sum_{k=0}^{K} (I_2(q_k))^2 - \mu_{I_2}(q_0)^2$$

are the variances of the two patches $p_0$ and $q_0$, respectively. Low values of SSD and SAD represents high photo-consistency, while NCC is the correlation between two image regions, therefore high values correspond to high photo-consistency. The computation of the NCC measure is more expensive, but the final score is resilient against linear illumination changes, and it partially compensates the limitation of the Lambertian model.
3.2.3 Depth Maps

Since the datasets of Seitz et al. (2006) and Strecha et al. (2008) were made available, MVS algorithms have been able to progress and vary according to the approach used, in some cases achieving very accurate results on these datasets. It is hard to provide a unique and complete classification of such as algorithms since the proposed methods differ over many aspects, e.g. input assumptions, the algorithm used to obtain the reconstruction, optimization methods. Seitz et al. (2006) proposed a subdivision according to the representation of the 3D model in depth maps, volumetric, level set and mesh methods. However MVS algorithms, especially the most complex and recent ones, often propose a pipeline requiring different representations of the scene at different stages. Due to the algorithms used in this thesis, we will focus the attention on depth maps methods.

A depth map is a particular image that stores for each pixel the depth of the scene. Usually, the approaches based on this representation involve two steps: reference depth map estimation and map merging. Given a set of images, for each pixel $i$ of a reference image $I_{\text{ref}}$, depth estimation aims at recovering the depth pixel, by pairwise comparing the appearance of the images. The basic idea is to look for the most likely depth $z_i$ generated from the $i^{th}$ pixel of camera $C_i$ with respect to the image in camera $C_j$, as shown in Figure 3.9.

![Figure 3.9: Depth estimation from camera $C_1$ of pixel $x_i$ w.r.t camera $C_2$](image)

Figure 3.9: Depth estimation from camera $C_1$ of pixel $x_i$ w.r.t camera $C_2
A very popular approach, known as stereo matching and exposed in the work of Scharstein and Szeliski (2002), compares two images as it follows: for each pixel \( i \) of the reference frame, the algorithm scans the corresponding epipolar line in a second image, and looks for the pixel whose neighbouring patch best matches the patch around \( i \). Commonly used matching costs are the SSD, described by Equation (3.17), or the NCC, described by Equation (3.19). Another approach, named plane-sweeping and designed by Collins (1996), scans the entire 3D scene, that needs to be discretized, with a fronto-parallel plane, i.e. a plane parallel to the reference image plane where the other images are projected. For each pixel \( i \) in the reference images, plane-sweeping compares the neighbourhood patch against the corresponding patches in the set of projected images, and choose the depth \( z_i \) corresponding to the best matching score. More recent approaches propose the usage of planes in multiple directions in the whole scene, like in the work realized by Gallup et al. (2007), or locally, like in the work realized by Sinha et al. (2014). Space sweeping approaches compare the image with more accuracy with respect to the previous method, thanks to the 3D projection, but the computational effort is much more expensive. The previous approaches output a noisy depth map that needs to be smoothed. An elegant approach to depth map estimation consists in minimizing the energy equation:

\[
E(z) = \sum_i \phi(z_i) + \sum_{ij} \psi(z_i, z_j) \tag{3.23}
\]

which combines both the function \( \phi \), encoding the photo-consistency described in the previous approaches, and a function \( \psi \) defined over the neighbourhood of the pixel \( i \) that penalizes differences, to encode a smoothness prior. The minimization of this energy usually bootstraps from the noisy depth maps and leads to more accurate and smooth estimations. Different optimization tools have been adopted to minimize Equation (3.23). In their work, Campbell et al. (2008) store multiple hypotheses for each pixel depth, estimated with a classical stereo matching algorithm, then they optimize this initial depth map by modelling the problem as a Markov Random Field.

There are some drawbacks affecting the depth map computations. For instance, estimated depth maps are usually noisy and their quality is extremely linked to the content of the image. Indeed, the depth of untextured regions is very challenging to model, such as the regions
along the boundaries of occluded objects where the foreground objects corrupt the stereo matching process. Some recent works address explicitly these issues, such as the work of Semerjian (2014) proposes a way to estimate the depth maps as an energy minimization problem that aims at producing an edge-preserving smooth depth maps, or the work of Wei et al. (2014) in which is exposed a coarse-to-fine approach together with the propagation of the depth belief that lead to cleaned depth maps.

3.2.4 PatchMatch-based and Learning-based MVS

Among all the possible approaches to reconstruct a 3D scene, the most performing ones are currently the MVS algorithms PatchMatch-based and Learning-based. In both cases, the main activity consists in the most possible accurate estimation of the depth map of each input image. Once computed, these maps are fused together into a dense point cloud as explained by Zheng et al. (2014a), or into a volumetric representation as Savinov et al. (2016), Blaha et al. (2016), Labatut et al. (2007), Vu et al. (2012) and Kuhn et al. (2017) do in their works.

The PatchMatch algorithm has been pioneered by Barnes et al. (2009) and improved by Bleyer et al. (2011) and by Zheng et al. (2014b) in their studies, and is based on the idea of choosing for each pixel a random guess of the depth and then propagate the most likely estimates to their neighbourhood, as shown in Figure 3.10. The work proposed by Schönberger et al. (2016), named COLMAP, can be con-

![Figure 3.10: The general PatchMatch scheme (From Zheng et al. (2014b))](image)
sidered the milestone of modern PatchMatch-based algorithms: it is a robust framework able to process high-quality images and to jointly estimate the pixel-wise camera visibility, and the depth and normal maps for each view. Since this method heavily relies on the Bilateral NCC photo-consistency, it often fails in recovering areas with low texture. Recently, to compensate for this drawback, TAPA-MVS has been designed by Romanoni and Matteucci (2019) to explicitly handle textureless regions. It propagates in a planar-wise fashion the valid depth estimate to neighbouring textureless areas. Kuhn et al. (2019) extends this method with a hierarchical approach improving the robustness of the estimation process.

Another family of MVS algorithm relies on Deep Learning. Deep-MVS by Huang et al. (2018) and MVSNet by Yao et al. (2018) were the first approaches proposing an effective MVS pipeline based on Neural Networks. For each camera, similarly to plane-sweeping designed by Collins (1996), they build up a cost volume by projecting nearby images on planes at different depths, then they classify or regress the best depth for each pixel, like in the works of Huang et al. (2018) and Yao et al. (2018), respectively. Subsequently, Yao et al. (2019) introduces in his work an RNN architecture to regularize the cost volume, while Luo et al. (2019) introduces a model to learn how to aggregate the cost to compute a more robust depth estimate. MVS-CRF model by Xue et al. (2019) refines the MVSNet estimate through Markov Random Field, while Point-MVSNet model by Chen et al. (2019) uses the recent graph-based neural architectures to achieve the same objective. The huge limitation of Learning-based approaches relies on their computational complexity. Usually, they are not able to handle images with high-resolution since both memory and time costs grow cubically as the volume resolution increases, causing a limitation on the accuracy of the reconstructed models. The best attempt to handle this problem is the work of Gu et al. (2019), named CasMVSNet in which, by applying a coarse-to-fine approach and by mapping feature according to the Feature Pyramid Network schema designed by Lin et al. (2017), they considerably improve the scalability of MVSNet methods.
### 3.2.5 COLMAP

COLMAP can be considered as the godfather of modern PatchMatch-based MVS algorithms. It is a model built on top of Zheng et al. (2014b) work, designed to produce a robust and efficient dense point cloud starting from an undistorted collection of images, thanks to its ability to jointly estimate the depth and normal maps and pixel-wise view selections according to geometric and photometric priors, that are subsequently fused together, as deeply described in the paper of Schönberger et al. (2016). Here we give an intuition of its functionalities in order to understand the content of the next chapters.

In accordance with the authors’ notation, we express the coordinates of an image pixel with the only value $l$, since the framework sweeps independently every single line of alternating between rows and columns. Given a reference image $X_{\text{ref}}$ and a set of source images $X_{\text{src}} = \{X^m | m = 1...M\}$ the framework computes the estimation of the depth $\theta_l$ as well as the normal $n_l$ for each pixel $l$ in parallel with the binary variable $Z_{m}^l \in \{0, 1\}$, which indicates if $l$ is visible in image $m$ or not. This is used to compute the Maximum-A Posteriori (MAP) estimation, where the posterior probability is:

$$ P(Z, \theta, N | X) = \frac{P(Z, \theta, N, X)}{P(X)} $$

$$(3.24)$$

where $L$ is the total number of pixels taken into account during the current line sweep, $X = \{X^{\text{src}}, X^{\text{ref}}\}$, $N = \{n_l | l = 1...L\}$ and $t = 1...T$ is the step in the optimization process. The following likelihood term:

$$ P(X^m_l | Z^m_l, \theta_l) = \begin{cases} \frac{1}{NA} \exp\left(-\frac{(1-\rho^m_l(\theta_l))^2}{2\sigma^2}\right) & \text{if } Z^m_l = 1 \\ \frac{1}{\pi} U & \text{if } Z^m_l = 0 \end{cases} $$

$$(3.25)$$

represents the photometric consistency of the patch $X^m_l$, which belongs to a non-occluded source image $m$ and is around the pixel corresponding to the point at $l$, with respect to the patch $X^{\text{ref}}_l$ around $l$ in the reference image. The photometric consistency $\rho$ is computed as a bilaterally weighted NCC, $A = \int_{-1}^{1} \exp\left\{-\frac{(1-\rho)^2}{2\sigma^2}\right\} d\rho$ and the constant $N$
cancels out in the optimization. The likelihood term $P(\theta_l, n_l|\theta_m^m, n_m^m)$ represents the geometric consistency and enforces multi-view depth and normal coherence. Finally $P(Z_{lm}^m|Z_{l-1,m}^m, Z_{l,t-1}^m)$ favors image occlusion indicators which are smooth both spatially and along the successive iteration of the optimization procedure.

Being Equation (3.24) intractable, Zheng et al. (2014b) proposed to use variational inference to approximate the real posterior with a function $q(Z, \theta, N)$ such that the KL divergence of the two functions is minimized. Schönberger et al. (2016) factorize $q(Z, \theta, N) = q(Z)q(\theta, N)$ and, to estimate such approximation, they propose a variant of the Generalized Expectation-Maximization algorithm propoded by Neal and Hinton (1998). In the E step, the values $(\theta, N)$ are kept fixed, and, in the resulting Hidden Markov Model, the function $q(Z_{ml,t}^m)$ is computed by means of message passing. In the M step, viceversa, the values of $Z_{ml,t}^m$ are fixed, the function $q(\theta, N)$ is constrained to the family of Kroneker delta functions $q(\theta_l, n_l) = q(\theta_l = \theta_l^*, n_l^*)$. The new optimal values of $\theta_l$ and $N_l$ are computed as:

$$
(\hat{\theta}_l^{opt}, \hat{n}_l^{opt}) = \arg\min_{\theta_l^*, n_l^*} \frac{1}{|S|} \sum_{m \in S} (1 - \rho_l^m(\theta_l^*, n_l^*))
$$

where $S$ is a subset of sources images, randomly sampled according to a probability $P_l(m)$ that favours images not occluded, and coherent with three priors which encourage good inter-cameras parallax, similar resolution and camera, front-facing the 3D point defined by $\theta_l^*, n_l^*$.

Figure 3.11: 3D dense reconstruction of the Todai-ji temple made through the use of the PatchMatch-based Multi-View Stereo algorithm by COLMAP (from Schönberger et al. (2016))
According to the PatchMatch scheme proposed by the authors, the pair \((\theta^l_i, n^l_i)\) evaluated in Equation (3.26) is chosen among the following set of hypotheses:

\[
(\theta^l_i, n^l_i), (\theta^{prp}_{i-1}, n^{prp}_{i-1}), (\theta^{rnd}_{i}, n^{rnd}_{i}), (\theta^{rand}_{i}, n^{rand}_{i}), (\theta^{prt}_{i}, n^{prt}_{i}), (\theta^{prt}_{i}, n^{prt}_{i})
\]  

(3.27)

where \((\theta_i, n_i)\) comes from the previous iteration, \((\theta_{i-1}, n_{i-1})\) is the estimate from the previous pixel of the scan, \((\theta^{rand}_{i}, n^{rand}_{i})\) is a random hypothesis and finally, \(\theta^{prt}_{i}\) and \(n^{prt}_{i}\) are two small perturbations of the estimates \(\theta_i\) and \(n_i\).

The main ingredient that makes COLMAP a successful MVS algorithm is the quality and the discriminative effectiveness of the stereo comparison among patches belonging to different cameras. Such comparison relies on a photometric measure, computed as NCC or similar metrics such as Sum of Squared Differences (SSD), or Bilateral Weighted NCC. We chose this algorithm for its ability to handle with efficiency both low-resolution and high-resolution image sets, making it the perfect candidate for our purposes. The major drawback arises in correspondence of untextured regions. Here the discriminative capabilities of NCC become unreliable because all the patches belonging to the untextured area are similar among each other. We will see how to handle this issue in the next chapter.

### 3.2.6 TAPA-MVS

TAPA-MVS is a novel PatchMatch-based MVS algorithm built on the shoulders of COLMAP proposed by Romanoni and Matteucci (2019). In addition to the fact it manages to obtain results comparable to its predecessor where this has excellent performance, TAPA-MVS is born to optimize the 3D reconstruction of all those areas without textures, i.e. those surfaces that the older PatchMatch-based algorithms they have difficulty to reconstruct accurately.

The idea behind this framework is to segment images into superpixels so that each superpixel would span a region of the image with a texture mostly homogeneous and it likely stops in correspondence to an image edge. Then, it propagates the depth and normal estimates belonging to photometrically stable regions around the edges to the entire superpixel. The algorithm contribution starts from the output of the first iteration of COLMAP, which is able to produce depth and
normal estimations only in correspondence of highly textured regions, as shown in Figure 3.12.

The idea of the method is to augment the set of PatchMatch depth hypotheses in Equation (3.27) with novel hypotheses that model a piecewise planar prior corresponding to untextured areas. In the first step the algorithm extracts the superpixels \( S = \{ s_1 \ldots s_{N_{\text{super}}} \} \) of each image by means of the algorithm SEEDS made by Van den Bergh et al. (2015). Since a superpixel \( s_k \) generally contains a homogeneous texture, the authors assume that each pixel covered by a superpixel \( s_k \) roughly belongs to the same plane. After running the first iteration of depth estimation, the algorithm filters out the small isolated speckles of the depth map obtained. As a consequence, the area of \( s_k \) in the filtered depth map likely contains a set \( \mathcal{P}_{\text{inl}}^{\kappa} \) of reliable 3D points estimates which roughly corresponds to real 3D points. In the presence of untextured regions, these points mostly belong to the areas near edges. Then, it fits a plane \( \pi_k \) on the 3D points in \( \mathcal{P}_{\text{inl}}^{\kappa} \) with RANSAC, classifying the points farther than 10 cm from the plane as outliers. Let \( \hat{\theta}_x \) be the tentative depth hypothesis for a pixel \( x \) corresponding to the 3D point on the plane \( \pi_k \) and let \( \hat{n}_x \) be the corresponding plane normal.

Then, let the inlier ratio be \( r_{\text{inl}}^{\kappa} = \frac{\text{num. inliers}}{|\mathcal{P}_{\text{inl}}^{\kappa}|} \), whose value expresses the confidence of the plane estimate. The actual hypotheses \( (\theta_x, n_x) \) for a pixel \( x \in s_k \) is generated as follows. To deal with fitting uncertainty,
the authors define \( P \left( (\theta_x, n_x) = (\hat{\theta}_x, \hat{n}_x) \right) = r_k^{\text{inl}} \); so that if the value \( v_{\text{ran}} \) sampled from a uniform distribution is \( v_{\text{ran}} < r_k^{\text{inl}} \) then \( \theta_x = \hat{\theta}_x \).

To propagate the hypotheses from superpixels with good inlier ratio to the neighbours with bad one, if \( v_{\text{ran}} > r_k^{\text{inl}} \) the value of \( \theta_x \) is sampled from the neighbouring superpixels belonging to a set \( N_k \). Since the aim is to spread the depth hypotheses among superpixels with a similar appearance, they sample from \( N_k \) proportionally to the Bhattacharya distance among the RGB histograms of \( s_k \) and the elements of \( N_k \). After several experiments, the authors choose to generate two depth hypothesis for each pixel \( N_{\text{fine}} = \frac{\text{imagewidth}}{20} \) and \( N_{\text{coarse}} = \frac{\text{imagewidth}}{30} \).

To integrate the novel hypotheses into the estimation framework, it is possible to simply add \((\theta_x^{\text{fine}}, n_x^{\text{fine}})\) and \((\theta_x^{\text{coarse}}, n_x^{\text{coarse}})\) to the set of hypotheses defined in Equation (3.27). However, in this case, these hypotheses would be treated with no particular attention to untextured areas. Indeed, the optimization framework would compare them against the baseline hypotheses relying on the photo-consistency metric; in the presence of flat evenly colored surfaces, the unreliability of the metric would still affect the estimation process. Instead, the goal of the proposed method is to favor \((\theta_x^{\text{fine}}, n_x^{\text{fine}})\) and \((\theta_x^{\text{coarse}}, n_x^{\text{coarse}})\) where the image presents untextured areas, so to guide the optimization to choose them instead of other guesses.

For these reasons, a pixel-wise textureness coefficient is defined to measure the amount of texture that surrounds a pixel \( x \). With a formulation similar to those presented by Vu et al. (2011), we define it as:

\[
t_x = \frac{\text{Var}_x + \epsilon_{\text{var}}}{\text{Var}_x + \frac{\epsilon_{\text{var}}}{t_{\text{min}}}}
\]

where \( \text{Var}_x \) is the variance of the 5x5 patch around pixel \( x \), \( \epsilon_{\text{var}} \) is a constant we fixed experimentally at 0.00005 and \( t_{\text{min}} = 0.5 \) is the minimum value we choose for the textureness coefficient; the higher the variance, the closer the coefficient is to 1.0.

To seamlessly integrate the novel hypotheses, the textureness coefficient is used to reweight the photometric-based cost \( C_{\text{photo}} = 1 - \rho(\theta, n) \) starting from Equation (3.26). Given a pixel \( x \) let define two weights:

\[
w^+(x) = 0.8 + 0.2 \cdot t_x \quad \text{and} \quad w^-(x) = 1.0 - 0.2 \cdot t_x.
\]

The metric \( C_{\text{photo}} = w^- \cdot C_{\text{photo}} \) is used for the hypotheses contained in the set of Equation (3.27) and \( C_{\text{photo}} = w^+ \cdot C_{\text{photo}} \) for \((\theta_x^{\text{fine}}, n_x^{\text{fine}})\) and \((\theta_x^{\text{coarse}}, n_x^{\text{coarse}})\) so that regions with low texture favors novel hypotheses. Vice-versa, it
Figure 3.13: Example of TAPA-MVS depth map refinement pipeline: the original depth map (a) is filtered to remove speckles (b) and finally is refined (c) (From Romanoni and Matteucci (2019)).

is better to force a higher geometric consistency $C_{geom}$ when we are dealing with the novel hypothesis in the presence of untextured areas. So to keep the formulation simple the authors use $w^+$ and $w^-$ again turning $\bar{C}_{geom} = w^+ \cdot C_{geom}$ for the standard set of hypotheses and $\bar{C}_{geom} = w^- \cdot C_{geom}$ for the proposed ones.

To ensure that the filtering of the maps obtained does not cut off all the estimates done so far that are not photometrically and geometrically consistent among the views, the authors insert in the estimation process a final depth refinement step. More in detail, they first apply a classical speckles filter to remove small blobs containing non-continuous depths values to consider two pixels as continuous when the depth difference is at most $10\%$ of the scene size. Then, to the regions where depth and normal estimates are missing, they designed the following refinement step, obtaining results similar to the content of Figure 3.13.

Let $x_{\text{miss}}$ be a pixel where depth and normal estimates are missing and $\mathcal{N}_{\text{miss}}$ the set of neighboring pixels. The simplest solution is to fill the missing estimate by averaging the depth and normal values contained in $\mathcal{N}_{\text{miss}}$. A better choice would be to weight the contribution to the average with the bilateral coefficients adopted in the bilateral NCC computation; they give more importance to the pixels close to $x_{\text{miss}}$ both in the image and in colour space. To better deal with depth discontinuities, the refinement process can be improved even further by using a weighted median of depth and normal instead of the weighted average. The pixel-wise median and, in particular, the weighted median is computational demanding, thus, to approximate the median computation, a three bins histogram is populated with the depths of the pixels in $\mathcal{N}_{\text{miss}}$. Finally, the bin with the highest frequency is chosen so to get rid of the outliers, and a bilaterally weighted average of
3.2. Multi-View Stereo

the depth and normal values that populate this bin is computed. The so obtained depth and normal values are assigned to \( x_{\text{miss}} \).

We choose this second PatchMatch-based framework for our experiments to have a better generalization of Super-Resolution behaviour applied to this family of MVS techniques. Moreover, we are interested in understanding how Super-Resolution untextured area are handled by an algorithm like TAPA-MVS.

3.2.7 CasMVSNet

CasMVSNet is a Deep Learning model designed by Gu et al. (2019) to efficiently produce 3D reconstructions through the use of a neural network architecture. It is a completely different approach to MSV, being a Deep Learning-based framework instead of a PatchMatch-based one. It is built over the MVSNet model, based on 3D CNNs and designed by Yao et al. (2018), that, like almost all other Deep Learning-based models for MVS, suffers from huge scalability problems linked to the complexity of the task to solve and is limited to low-resolution input images. The great contribution of this work lies in the fact of being able to reconstruct 3D scenes not only obtaining very thick dense point clouds as a result but also in the fact of being able to start from high-resolution input images without requiring, for this reason, huge computational availability.

Generally, Learning-based MVS pipelines compute the so-called 3D cost volume to quantify similarities between image patches and determine if they have to be matched or not. To do this, they compute disparity planes, then warp the extracted 2D features of each view to the hypothesis planes to build the feature volumes through the use of differentiable homography as follows:

\[
H_i(d) = K_i \cdot R_i \cdot (I - \frac{(t_1 - t_i) \cdot n_1^T}{d}) \cdot R_i^T \cdot K_i^{-1}
\] (3.29)

where \( H_i(d) \) is the homography between the feature map of the \( i^{th} \) view and the reference feature maps at depth \( d \), \( K_i, R_i \) and \( t_i \) are camera intrinsics, camera rotations and camera translations of the \( i^{th} \) view, respectively, and \( n_1 \) is the principal axis of the reference camera. Finally, they fuse them together to obtain the 3D cost volume, according to a variance-based metric.

The cost volume is a hypercube with dimensions \( W \times H \times D \times F \),
where $W \times H$ are the input images dimensions, $D$ is the number of hypothesis planes and $F$ is the channel number for each feature map. By increasing the spatial resolution $W \times H$ as well as $D$ the result is much more accurate but requires more memory to be computed. The authors reduce the hypothesis range by refining it over many stages in an iterative way. Let $R_k$ be the hypothesis range and $w_k < 1$ the reducing factor at $k^{th}$ stage. Starting from $R_1$, which covers the entire disparity range, CasMVSNet computes the next range predicting it from the previous, so that $R_{k+1} = R_k \cdot w_k$. At the same way, the hypothesis plane interval $I_k$ at $k^{th}$ stage, with its reducing factor $p_k < 1$, is updated as $I_{k+1} = I_k \cdot p_k$ to obtain more detailed outputs. At $k^{th}$ stage, there is a number of planes $D_k = \frac{R_k}{I_k}$ that is efficiently reduced stage by stage according to the aforementioned formulation.

As in the Feature Pyramid Network designed by Lin et al. (2017), the model doubles the spatial resolution stage by stage to reach input dimensions only at the last iteration, as can be seen in Figure 3.14. By calling $N = 3$ the maximum number of cascade cost volume stages, the spatial resolution at $k^{th}$ iteration is defined as $\frac{W}{2^{N-k}} \times \frac{H}{2^{N-k}}$. It is now possible to rewrite the homography warming function at the $(k+1)^{th}$ stage as follows:

$$H_i(d^m_k + \Delta^m_{k+1}) = K_i \cdot R_i \cdot (I - \frac{(t_1 - t_i) \cdot n_1^T}{d^m_k + \Delta^m_{k+1}}) \cdot R_i^T \cdot K_i^{-1} \quad (3.30)$$

where $d^m_k$ is the predicted disparity of $m^{th}$ pixel at the $k^{th}$ stage and
\( \Delta_{k+1}^m \) is the residual disparity of \( m^{th} \) pixel at the \( k^{th} \) stage. Instead of generating several low-resolution disparity maps from a single fixed cost volume, as explained by Chang and Chen (2018), Guo et al. (2019), Kendall et al. (2017) and Zhang et al. (2019), to then upsampling and refining them, in order to obtain high-resolution disparity maps the authors build three cost volumes from the feature maps of the Feature Pyramid Network with corresponding spatial resolution equal to \( \frac{1}{16}, \frac{1}{4}, 1 \) of the input images size, respectively. Finally, the loss functions of the intermediate stages are weighted and summed to obtain the final loss of the entire model:

\[
L_{\text{final}} = \sum_{k=1}^{N} \lambda^k \cdot L^k
\]

where \( L^k \) is the loss at \( k^{th} \) stage and \( \lambda^k \) is its corresponding weight. We choose this model because of its scalability, that let us able to reconstruct both low-resolution and high-resolution scenes, and also to understand Super-Resolution behaviour applied in top of a Learning-based MVS algorithm.
Chapter 4

Super-Resolution Multi-View Stereo

The aim of this thesis is to investigate whether, how and how much Single-Image Super-Resolution (SISR) algorithms can improve the 3D reconstruction of a scene. In the literature, several works exploited SISR to improve the quality of 3D models from different perspectives.

In his work Goldlücke et al. (2014) proposes a variational method to improve the appearance by estimating Super-Resolution textures on a 3D model. Another model based on SISR is proposed by Maier et al. (2015) and relies on the idea of fusing multiple RGB-D and colour images into Super-Resolution depth maps and RGB keyframes to enhance the texture mapping process. Instead, Hui et al. (2016), Zuo et al. (2019) and Voynov et al. (2019) proposed in their works different models to exploit high-resolution RGB images to guide a DeepLearning-based architectures to increase depth maps resolution.

In our work, we aim to improve the performance of 3D reconstructions based on Multi-View Stereo techniques through the use of the SISR directly on the input images. For this purpose, we use several powerful and modern algorithms provided by the Computer Vision community, with the intention of classifying the scenarios and cases in which Super-Resolution can lead to better results than the basic models. In addition to this, we are interested in investigating differences and similarities in the behaviours of the PatchMatch-based and DeepLearning-based algorithms proposed in response to the same Super-Resolution algorithm placed at the input. Since this is an experimental thesis, our main contribution focuses on the section dedicated
to experiments. We will use almost the rest of the document to thoroughly examine and validate our experiments to ensure that the results can be as general and accurate as possible. We believe that increasing the size of the input through a Super-Resolution algorithm can both improve the estimate of the depth maps given the greater quantity and quality of details, and enhance the completeness of the reconstructed model, increasing the number of points that make it up. As far as we know, at the time of this writing, there are no documents or projects that have tested this type of approach.

Let $\mathcal{S}$ be a set of $N$ sequences $\mathcal{I}_n$, each one composed by images from the same scene captured by one or more cameras. Let $\mathcal{MVS}$ be a generic Multi-View Stereo pipeline able to process high-resolution images, and $\mathcal{SR}^k$ a SISR function with scale factor $k$. For each sequence $\mathcal{I}_n$, having that $w_n$ and $h_n$ are the dimensions of its images, we apply the SISR function obtaining:

$$I_n^k = \mathcal{SR}_k(\mathcal{I}_n)$$
$$w_n^k = k \cdot w_n$$
$$h_n^k = k \cdot h_n$$

(4.1)

We call $\mathcal{S}^k$ the set of these new sequences. At this point, we compute the camera parameters $\mathcal{C}_n^k$ for each sequence of the new set, and we use them to compute the sparse point cloud and to undistort the Super-Resolution sequences, obtaining $\mathcal{U}_n^k$. Then, we feed the MVS algorithm with each undistort the Super-Resolution sequence and we obtain the sequences of Super-Resolution maps $\mathcal{M}_n^k = \mathcal{MVS}(\mathcal{U}_n^k)$. Finally, after filtering the obtained outputs, if required according to the MVS algorithm used, we fuse together the obtained maps obtaining the Super-Resolution dense point cloud, as shown in Figure 4.1.

In order to obtain the best possible reconstruction, for our experiments, we use a Super-Resolution function with a scale factor $k = 2$. It is in fact proven that increasing the scale factor also increases the difficulty of the problem to be solved since the algorithm must predict a greater number of pixels starting always from the same low-resolution image. In analytical terms, this translates into a drop in performance in terms of PSNR and SSIM. So if we increase the scale factor we will obtain larger images, and therefore the reconstruction algorithm would have a greater amount of information available from which to extract the 3D model. However, it is also true that, since these images are very
In our experiments, we will always use the Deep Back-Projection Network (DBPN) SISR model designed by Haris et al. (2019). The authors contributed both by implementing various versions of the model and by providing the weights calculated by them. We chose the version with residual learning with multiple pairs of projection units which is, according to authors results, the most performing with each scale factor.

First of all, we want to evaluate the contribution of the chosen SISR function put on top of PatchMatch-based MVS pipelines. As deeply explained in the previous chapter, they are currently among the most successful MVS approaches especially since they can smoothly handle high-resolution images and fully exploit all the information contained in them. In this thesis we focus mainly on the study of the behaviour of COLMAP, a model created by Schönberger et al. (2016), having within its pipeline what can be considered the father of modern PatchMatch-based MVS algorithms. Knowing of the low performance that this family of algorithms has in regions without textures, caused by a photometric consistency based on NCC, we expect to see worsening in Super-Resolution regions of this kind.

To try to cope with this possibility, we intend to relax the fundamental parameters of this PatchMatch-based algorithm, that is, as sug-

![Diagram](image.png)

*Figure 4.1: The general pipeline of the proposed model. Given a set of initial images, these are processed by the Super-Resolution algorithm, after which it is fed to the Structure from Motion pipeline which calculates the parameters of the associated cameras. Finally, these parameters are used to produce the undistorted images that will be fed to the chosen Multi-View Stereo algorithm, which will return the Super-Resolution version of the dense point cloud.*
gested by the authors, by reducing the minimum threshold in terms of NCC and by increasing the maximum window radius of the patches for the same scale factor $k = 2$ chosen for SISR step, and to filter the depth maps thus obtained, which we expect to be noisy, with the filterSpeckles algorithm. This type of algorithm is based on parallelepiped-shaped volumes, with base area $s$ and depth $d$, which flow iteratively along the map to be filtered by cutting out all those pixels inside the area $s$ with a depth greater than $d$. After a series of grid search experiments aimed at maximizing the average F1, we decided to fix the maximum size of the speckles $s_{\text{max}}$ and their maximum depth $d_{\text{max}}$ so that:

$$s_{\text{max}} = \frac{w_M \cdot h_M}{2500}$$
$$d_{\text{max}} = \frac{1}{2}$$

where $w_M$ and $h_M$ are the width and the height of the input map, respectively.

In addition, we combine COLMAP with one of the latest PatchMatch-based algorithm called TAPA-MVS and developed by Romanoni and Matteucci (2019) in order to reconstruct these regions lacking in texture details with much more efficiency. In this case, we choose to use the algorithm with its default settings without changing neither the parameters for the estimation of the depth maps nor for the filtering phase, as they constitute one of the substantial differences with COLMAP’s PatchMatch algorithm.

Finally, in order to generalize and get a more accurate idea of the effects of Super-Resolution on 3D reconstructions, we test the behaviour of the DBPN algorithm on top of CasMVSNet, the MVS DeepLearning-based scalable algorithm designed by Gu et al. (2019). In fact, we want to understand the differences and similarities between this approach and the two previously presented PatchMatch-based. In agreement with the authors’ suggestions, given the complexity of the network and the difficulties they expressed in making it converge, we use the weights provided by them with the parameters used in the training phase. These parameters also include the maximum dimensions of the cost volume, i.e. the maximum width and height of the hypercubes associated with each image of each sequence during the reconstruction phase, set by default to $1152 \times 864$. These dimensions should be adapted to those of the input images, especially in the case of the
Super-Resolution ones in order to take advantage of the huge increase in pixels, due to the fact that the algorithm proportionally rescales all the larger images of this size.

The main reason why we decided not to increase these limits is the required computing power, which grows cubically. Furthermore, the sequences that we will present in the next chapter have a great variance in the number of images, and in some cases, even a slight increase in maximum dimensions is sufficient to saturate the computing resources. Finally, to compensate for this limitation, and to be able to reconstruct even more accurately scenes with great depth, we reduced the interval scale from 1.06 to 0.9 as suggested by the contributors via the Github platform. This parameter is inversely proportional to the depths of the hypercube and represents the divisor in calculating the number of hypothesis planes.

For each of the three MVS algorithms chosen, we will reconstruct both the basic-resolution sequences and the Super-Resolution sequences. After that, in accordance with the tolerances $\tau$ associated with each set and experienced in centimetres, we will perform the evaluation of the reconstructions obtained with respect to the model obtained with the laser, which we will consider to be the ground truth.
Chapter 5

Experiments and Results

There are several benchmarks built with the purpose of training and evaluating Multi-View Stereo (MVS) algorithms. Among the most famous we surely have to mention Tanks and Temples made by Knapitsch et al. (2017), DTU-MVS made by Jensen et al. (2014) and ETH3D made by Schops et al. (2017). Because of computational limitations due to both the Super-Resolution algorithm and the MVS algorithms, we are forced to base our experiments on the low-resolution many-view version of the ETH3D benchmark. In fact, applying the proposed pipeline to the other benchmarks would mean reconstructing scenes starting from 4K images, whose computation is extremely expensive.

The low-resolution many-view ETH3D benchmark, called from now on $S$, is made up of 10 image datasets, 5 belonging to the Train set $S_{\text{train}}$ and 5 belonging at Test set $S_{\text{test}}$. Each of these subsets is in turn divided into 3 Outdoor sequences $S_{\text{out}}$ and 2 Indoor sequences $S_{\text{in}}$.

Furthermore, we apply a qualitative subdivision, more suitable for the purposes of our experiments, based on the content of the sequences, i.e. subdividing them into Textured sequences $S_{\text{txd}}$ and Textureless sequences $S_{\text{txl}}$, according to the massive presence of details and shapes with defined edges in the first case, and according to the non-negligible presence of monochromatic, reflective or poor in details surfaces, such as walls or floors, in the second case. More in detail, we are going to consider delivery_area, electro, terrains in Textureless train sequences $S_{\text{train,txl}}$, forest, playground in Textured train sequences $S_{\text{train,txd}}$, storage_room, storage_room_2 in Textureless test sequences $S_{\text{test,txl}}$ and lakeside, sand_box, tunnel in Textured train sequences $S_{\text{test,txd}}$. 

77
For each sequence $I$ the authors of the benchmark provide an already estimated set of Pinhole camera model parameters $C$. According to the math behind this camera model, to compute their equivalent for each Super-Resolution set $I^2$, we simply multiply the intrinsic parameters matrix for scale factor $k = 2$, and by this way we obtain $C^2$. Then, for each sequence $I$ and $I^2$, taking advantage of the starting images and with the parameters of the associated chambers $C$ and $C^2$, we apply, independently of the MVS model under analysis, the image undistorter algorithm of COLMAP to obtain the sets of undistorted images $U$ and $U^2$ used to let reconstructions execute.

The result evaluation is computed in two steps: first, the reconstructed scene is aligned with its ground truth, then each estimated 3D point is evaluated as correct if its distance from its twin belonging to the ground truth is lower than a threshold, as wrong in the other case. This threshold is more properly called tolerance $\tau$, and it is composed of a set of values, in the case of ETH3D $\tau \in \{1\text{cm}, 2\text{cm}, 5\text{cm}, 10\text{cm}, 20\text{cm}, 50\text{cm}\}$, which are sequentially used to arrive at a more detailed series of results calculated on different scales. The metrics used to evaluate this benchmark are accuracy, completeness and the F1 score. Accuracy is defined as the fraction of reconstruction points which are within a distance threshold of the ground truth points. To determine completeness, we measure the distance of each ground truth 3D point to its closest reconstructed point. Completeness is defined as the fraction of ground truth points for which this distance is below the evaluation threshold. Finally, F1 is defined as the harmonic average between the two previous metrics, and as such, we will consider it as the main metric in our experiments.

All the experiments are executed over an Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz with an Nvidia GTX 1080Ti.
5.1 COLMAP-SR

In this section, we present the experiments carried out and the results obtained using COLMAP, the PatchMatch-based algorithm developed by Schönberger et al. (2016). What we expect from this algorithm is that it takes advantage of the pixel increase provided by Super-Resolution, managing not only to produce 3D reconstructions visually richer in details but also to increase the scores associated with their evaluations. Furthermore, we expect to see some slight deterioration in Textureless sequences, given the nature of the PatchMatch algorithm on which the MVS of this pipeline is based.

We will initially observe the behaviour of COLMAP on $S_{train}$ and $S_{train}^2$ sequences, comparing each result obtained according to the proposed metrics, and evaluating based on these if it is necessary or not to modify the parameters of the algorithm. Finally, we will validate the model on $S_{test}$ and $S_{test}^2$ sequences.

5.1.1 ETH3D train

As a first experiment, we reconstruct all the sequences $S_{train}$ and $S_{train}^2$ using COLMAP with its default parameter setting, naming this first model $COLMAP_{V0}$.

From the charts in Figure 5.1 it is possible to state that the Super-Resolution impacts positively the model in terms of completeness. Apparently, this algorithm reacts to the larger amount of pixel in input images with respect to the base-resolution model by creating denser point clouds. The general drop in accuracy, especially if low tolerances are considered, results in noise inside the reconstructions. So overall, the Super-Resolution reconstructions obtained are richer in points with respect to base-resolution ones, but in some cases, some of these are not positioned correctly.

Benefits and disadvantages are very evident also going to analyze the various subsets, which offer us a more detailed view. It is, in fact, possible to note, also by observing the data in Table 5.1, the great difference between Indoor and Outdoor sequences, as well as Textureless and Textured ones due to the effects of Super-Resolution. Textureless sequences are subject to a lot of noise, similarly albeit less incisively than Indoor sequences, and in addition, there is almost no contribution given by the increase in pixel amount. On the contrary, Textured
Figure 5.1: COLMAPV0 performance over $S_{train}$ and $S_{train}^2$. F1, accuracy and completeness are computed according with tolerances $\tau = 1, 2, 5, 10, 20, 50$cm. Red-line set at 0 represents the basic-resolution model to beat, while the interpolated points represent the difference between the performance of the Super-Resolution model and the aforementioned basic-resolution one, for each metric and for each tolerance.
Table 5.1: F1 scores over $S_{train}$ and $S_{2train}$ with COLMAP$_V^0$. We compare the performance of the Super-Resolution model against those of the basic-resolution one according with the tolerances $\tau$.

<table>
<thead>
<tr>
<th>$\tau$ (cm)</th>
<th>Overall</th>
<th>Indoor</th>
<th>Outdoor</th>
<th>Textureless</th>
<th>Textured</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>base SR</td>
<td>base SR</td>
<td>base SR</td>
<td>base SR</td>
<td>base SR</td>
</tr>
<tr>
<td>1</td>
<td>35.81</td>
<td>37.47</td>
<td>40.69</td>
<td>37.83</td>
<td>32.55</td>
</tr>
<tr>
<td>2</td>
<td>53.42</td>
<td>52.71</td>
<td>56.15</td>
<td>52.29</td>
<td>51.59</td>
</tr>
<tr>
<td>5</td>
<td>72.16</td>
<td>71.37</td>
<td>74.36</td>
<td>71.29</td>
<td>70.70</td>
</tr>
<tr>
<td>10</td>
<td>81.83</td>
<td>81.74</td>
<td>83.98</td>
<td>82.34</td>
<td>80.40</td>
</tr>
<tr>
<td>20</td>
<td>88.99</td>
<td>89.33</td>
<td>91.50</td>
<td>90.90</td>
<td>87.31</td>
</tr>
<tr>
<td>50</td>
<td>95.30</td>
<td>95.73</td>
<td>97.33</td>
<td>97.30</td>
<td>93.94</td>
</tr>
</tbody>
</table>

sequences tend to be much richer in points, and this can be seen from the massive increase in completeness that we also find in Outdoor sequences.

These results confirm the expectations for COLMAP’s PatchMatch-based MVS algorithm. Relying on a photometric consistency calculated as bilaterally weighted NCC, it was presumed that the performance would deteriorate in the presence of Textureless areas.

As described in the previous chapter, to cope with this behavioural inefficiency, we relax the algorithm parameters related to the PatchMatch phase. As an obvious consequence of this change, the depth maps produced are characterized by a large presence of noise, which would undoubtedly generate very inaccurate reconstructions. For this reason, we decided to filter these depth maps thus obtained through the filterSpeckles algorithm as explained in the previous chapter, obtaining maps with much more homogeneous depth values as can be seen from the example in Figure 5.2.

Let COLMAP$_{V1}$ be the COLMAP$_{V0}$ version with the relaxed PatchMatch explained so far without any filtering, and COLMAP$_{V2}$ the filtered version of COLMAP$_{V1}$. What can be seen from the results in Table 5.2 is that both the proposed filtering worsen the performance of the reconstruction with basic-resolution compared to COLMAP$_{V0}$. Instead, what we see from the comparisons between the Super-Resolution reconstructions in Table 5.3 is that both COLMAP$_{V1}$ and COLMAP$_{V2}$ worsen the performance of the Outdoor and Textured reconstructions, while improving them in the Indoor and Textureless scenarios.

In more detail, COLMAP$_{V2}$ results to greatly improve the F1 of
Table 5.2: F1 scores over $S_{train}$ according with the tolerances $\tau$. We compare the performance of COLMAP$_V^0$ against COLMAP$_V^1$ and COLMAP$_V^2$.

<table>
<thead>
<tr>
<th>$\tau$ (cm)</th>
<th>Overall</th>
<th>Indoor</th>
<th>Outdoor</th>
<th>Textureless</th>
<th>Textured</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0</td>
<td>V1</td>
<td>V2</td>
<td>V0</td>
<td>V1</td>
<td>V2</td>
</tr>
<tr>
<td>1</td>
<td>35.81</td>
<td>32.86</td>
<td>32.53</td>
<td>40.69</td>
<td>40.35</td>
</tr>
<tr>
<td>2</td>
<td>53.42</td>
<td>50.15</td>
<td>49.33</td>
<td>56.15</td>
<td>55.28</td>
</tr>
<tr>
<td>5</td>
<td>72.16</td>
<td>68.82</td>
<td>67.18</td>
<td>74.36</td>
<td>72.68</td>
</tr>
<tr>
<td>10</td>
<td>81.83</td>
<td>78.80</td>
<td>76.66</td>
<td>83.98</td>
<td>82.29</td>
</tr>
<tr>
<td>20</td>
<td>88.99</td>
<td>86.60</td>
<td>84.31</td>
<td>91.50</td>
<td>90.29</td>
</tr>
<tr>
<td>50</td>
<td>95.30</td>
<td>93.81</td>
<td>92.42</td>
<td>97.33</td>
<td>96.47</td>
</tr>
</tbody>
</table>

Table 5.3: F1 scores over $S_{2\text{train}}$ according with the tolerances $\tau$. We compare the performance of COLMAP$_V^0$ against COLMAP$_V^1$ and COLMAP$_V^2$.

<table>
<thead>
<tr>
<th>$\tau$ (cm)</th>
<th>Overall</th>
<th>Indoor</th>
<th>Outdoor</th>
<th>Textureless</th>
<th>Textured</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0</td>
<td>V1</td>
<td>V2</td>
<td>V0</td>
<td>V1</td>
<td>V2</td>
</tr>
<tr>
<td>1</td>
<td>37.47</td>
<td>37.63</td>
<td>38.08</td>
<td>37.83</td>
<td>41.66</td>
</tr>
<tr>
<td>2</td>
<td>52.71</td>
<td>52.53</td>
<td>53.44</td>
<td>52.29</td>
<td>55.81</td>
</tr>
<tr>
<td>5</td>
<td>71.37</td>
<td>70.59</td>
<td>70.96</td>
<td>71.29</td>
<td>73.58</td>
</tr>
<tr>
<td>10</td>
<td>81.74</td>
<td>80.90</td>
<td>80.73</td>
<td>82.34</td>
<td>83.84</td>
</tr>
<tr>
<td>20</td>
<td>89.33</td>
<td>88.51</td>
<td>88.10</td>
<td>90.90</td>
<td>91.73</td>
</tr>
<tr>
<td>50</td>
<td>95.73</td>
<td>94.99</td>
<td>95.07</td>
<td>97.30</td>
<td>97.27</td>
</tr>
</tbody>
</table>

the latter especially for low $\tau$ values, that is, with regard to the more severe assessments. These results translate graphically into Textureless areas with some holes but with much denser and more accurate points than those produced using COLMAP$_V^0$.

In light of these results, we have decided to divide the COLMAP pipeline into two different strands according to the sequence $I_2$ to be reconstructed. In particular:

$$COLMAP = \begin{cases} 
\text{COLMAP}_V^0 & \text{if } I_2 \in S_{txd}^2 \\
\text{COLMAP}_V^2 & \text{if } I_2 \in S_{txl}^2 
\end{cases} \quad (5.1)$$

while we keep $COLMAP = COLMAP_V^0$ for reconstructions with base-resolution.

At this point, we reconstruct all the sequences $S_{train}$ and $S_{2\text{train}}$ according to this new pipeline.

By comparing the graphs in the Figure 5.3 with those previously obtained and shown in the Figure 5.1, it is possible to notice a general
increase in all performance, such as to have F1 and completeness of the Super-Resolution reconstructions always better than those in the basic-resolution, with also a not negligible increase in accuracy. Going to analyze the target of these changes, i.e. the Textureless sequences, we note that all the metrics have enjoyed the changes made to the model, up to transforming the difference of the F1 score from worsening into improvement for $\tau = 1, 2$ cm.

Table 5.4: F1 scores over $S_{\text{train}}$ and $S_{\text{train}}^2$ with COLMAP. We compare the performance of the Super-Resolution model against those of the basic-resolution one according with the tolerances $\tau$.

<table>
<thead>
<tr>
<th>$\tau$ (cm)</th>
<th>Overall</th>
<th>Indoor</th>
<th>Outdoor</th>
<th>Textureless</th>
<th>Textured</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>35.81</td>
<td>40.00</td>
<td>43.42</td>
<td>32.55</td>
<td>37.71</td>
</tr>
<tr>
<td>2</td>
<td>53.42</td>
<td>54.98</td>
<td>56.15</td>
<td>51.59</td>
<td>53.58</td>
</tr>
<tr>
<td>5</td>
<td>72.16</td>
<td>72.64</td>
<td>74.36</td>
<td>70.70</td>
<td>71.99</td>
</tr>
<tr>
<td>10</td>
<td>81.83</td>
<td>82.13</td>
<td>83.98</td>
<td>80.40</td>
<td>81.77</td>
</tr>
<tr>
<td>20</td>
<td>88.99</td>
<td>89.14</td>
<td>91.50</td>
<td>87.31</td>
<td>88.50</td>
</tr>
<tr>
<td>50</td>
<td>95.30</td>
<td>95.70</td>
<td>97.33</td>
<td>93.94</td>
<td>94.75</td>
</tr>
</tbody>
</table>
Chapter 5. Experiments and Results

Figure 5.3: COLMAP performance over \( S_{\text{train}} \) and \( S_{\text{train}}^2 \). F1, accuracy and completeness are computed according with tolerances \( \tau = 1, 2, 5, 10, 20, 50 \) cm. Redline set at 0 represents the basic-resolution model to beat, while the interpolated points represent the difference between the performance of the Super-Resolution model and the aforementioned basic-resolution one, for each metric and for each tolerance.
5.1.2 ETH3D test

We evaluate the performance of the COLMAP model on $S_{test}$ and $S^2_{test}$, leaving its setup unchanged compared to the last evaluation on $S_{train}$ and $S^2_{train}$.

Both from the data in Table 5.5 and from the graphs in Figure 5.4 it is possible to realize that, except for a slight bias, the model performance confirm the anticipated trend on the train set. We can observe also, in this case, the effectiveness of the changes made on the basic model to face the problem of Textureless sequences, while for Textured ones we notice a general improvement, especially for low tolerances. In more detail, COLMAP manages to exploit the increase in pixels in the starting images to produce more complete reconstructions characterized by a slight presence of noise.

As can be seen from the reconstructions in Appendix A, this phenomenon is very evident in scenes rich in texture and often results in an increase in the depth of the reconstructed scene. As for scenes without textures or characterized by the presence of large smooth areas, COLMAP uses the effects of Super-Resolution to enrich the regions around their edges with correctly positioned points, however creating holes in the texture or mistaking the estimate of the depth in their central areas.

However, it is clear that the algorithm is not only able to cope with the presence of artifacts, but that thanks to Super-Resolution it is on average capable of improving both the analytical and visual quality of its reconstructions.

Table 5.5: F1 scores over $S_{test}$ and $S^2_{test}$ with COLMAP. We compare the performance of the Super-Resolution model against those of the basic-resolution one according with the tolerances $\tau$. In this case the Indoor sequences coincide with the Textureless ones, while the Outdoor sequences coincide with the Textured ones.

<table>
<thead>
<tr>
<th>$\tau$ (cm)</th>
<th>Overall</th>
<th>Textureless</th>
<th>Textured</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>base SR</td>
<td>base SR</td>
<td>base SR</td>
</tr>
<tr>
<td>1</td>
<td>37.35</td>
<td>28.09</td>
<td>43.52</td>
</tr>
<tr>
<td></td>
<td>41.20</td>
<td>29.57</td>
<td>48.95</td>
</tr>
<tr>
<td>2</td>
<td>55.51</td>
<td>44.97</td>
<td>62.53</td>
</tr>
<tr>
<td></td>
<td>56.21</td>
<td>44.15</td>
<td>64.25</td>
</tr>
<tr>
<td>5</td>
<td>74.23</td>
<td>65.63</td>
<td>79.96</td>
</tr>
<tr>
<td></td>
<td>73.53</td>
<td>63.77</td>
<td>80.03</td>
</tr>
<tr>
<td>10</td>
<td>83.59</td>
<td>77.50</td>
<td>87.65</td>
</tr>
<tr>
<td></td>
<td>83.27</td>
<td>76.23</td>
<td>87.97</td>
</tr>
<tr>
<td>20</td>
<td>89.92</td>
<td>86.37</td>
<td>92.28</td>
</tr>
<tr>
<td></td>
<td>90.16</td>
<td>85.79</td>
<td>93.08</td>
</tr>
<tr>
<td>50</td>
<td>95.42</td>
<td>94.03</td>
<td>96.35</td>
</tr>
<tr>
<td></td>
<td>96.07</td>
<td>94.39</td>
<td>97.19</td>
</tr>
</tbody>
</table>
Figure 5.4: COLMAP performance over $S_{test}$ and $S_{test}^2$. $F_1$, accuracy and completeness are computed according with tolerances $\tau = 1, 2, 5, 10, 20, 50$cm. Redline set at 0 represents the basic-resolution model to beat, while the interpolated points represent the difference between the performance of the Super-Resolution model and the aforementioned basic-resolution one, for each metric and for each tolerance.
5.2 TAPA-MVS-SR

In this section, we present the experiments carried out and the results obtained using TAPA-MVS, the PatchMatch-based algorithm developed by Romanoni and Matteucci (2019). What we expect from this algorithm is to see behaviours similar to those found with COLMAP in texture-rich reconstructions, with significant improvements in terms of completeness in exchange for a slight deterioration in accuracy. We are interested in finding out how the superpixel hypothesis reacts to Super-Resolution sequences with regions without textures, and whether this difference with COLMAP can lead to benefits in terms of performance for our experiments. We will initially observe the behaviour of TAPA-MVS on $S_{\text{train}}$ and $S_{\text{train}}^2$ sequences, comparing each result obtained according to the proposed metrics and evaluating on the basis of these if it is necessary or not to modify the parameters of the algorithm. Next, we will validate the model on $S_{\text{test}}$ and $S_{\text{test}}^2$ sequences.

5.2.1 ETH3D train

Like we did for COLMAP, we reconstruct all the sequences $S_{\text{train}}$ and $S_{\text{train}}^2$ but this time using TAPA-MVS with its default parameters setup.

From the charts in Figure 5.5 it is possible to state that the Super-Resolution impacts positively the model in terms of completeness for low tolerances, and in terms of accuracy for higher ones. Apparently, this algorithm reacts to the larger amount of pixels in input images with respect to the base-resolution model in two completely different ways according to the quality of the sequence in terms of texture.

Benefits and disadvantages are very evident going to analyze the various subsets, which provide us with a more detailed view. It is, in fact, possible to note, also by observing the data in Table 5.6, the great difference between Indoor and Outdoor sequences, as well as Textureless and Textured ones due to the effects of Super-Resolution. The Input sequences are subject to a reduction in noise proportional to the increase in tolerance, due to a clear reduction in the number of points considered by the algorithm during the fusion phase, while the sequences classified as Textureless follow this same trend only starting from medium tolerances. On the contrary, the Textured sequences
Table 5.6: F1 scores over $S_{train}$ and $S_{train}^2$ with TAPA-MVS. We compare the performance of the Super-Resolution model against those of the basic-resolution one according with the tolerances $\tau$.

<table>
<thead>
<tr>
<th>$\tau$ (cm)</th>
<th>Overall</th>
<th>Indoor</th>
<th>Outdoor</th>
<th>Textureless</th>
<th>Textured</th>
</tr>
</thead>
<tbody>
<tr>
<td>base SR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>38.87</td>
<td>42.33</td>
<td>45.22</td>
<td>44.67</td>
<td></td>
</tr>
<tr>
<td></td>
<td>34.64</td>
<td>40.77</td>
<td>41.61</td>
<td>43.73</td>
<td>34.77</td>
</tr>
<tr>
<td></td>
<td>55.13</td>
<td>56.09</td>
<td>58.20</td>
<td>56.78</td>
<td></td>
</tr>
<tr>
<td></td>
<td>53.07</td>
<td>55.62</td>
<td>57.73</td>
<td>57.22</td>
<td>51.21</td>
</tr>
<tr>
<td></td>
<td>72.54</td>
<td>72.78</td>
<td>73.59</td>
<td>72.15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>71.84</td>
<td>73.20</td>
<td>74.36</td>
<td>73.16</td>
<td>69.81</td>
</tr>
<tr>
<td></td>
<td>81.65</td>
<td>82.41</td>
<td>82.34</td>
<td>81.95</td>
<td></td>
</tr>
<tr>
<td></td>
<td>81.19</td>
<td>82.72</td>
<td>82.50</td>
<td>82.27</td>
<td>80.38</td>
</tr>
<tr>
<td></td>
<td>88.45</td>
<td>89.65</td>
<td>90.36</td>
<td>87.55</td>
<td></td>
</tr>
<tr>
<td></td>
<td>87.55</td>
<td>89.18</td>
<td>89.62</td>
<td>88.86</td>
<td>87.84</td>
</tr>
<tr>
<td></td>
<td>95.10</td>
<td>96.14</td>
<td>96.40</td>
<td>97.59</td>
<td></td>
</tr>
<tr>
<td></td>
<td>94.24</td>
<td>95.17</td>
<td>95.07</td>
<td>96.30</td>
<td>95.15</td>
</tr>
<tr>
<td></td>
<td>95.10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Not only tend to be much richer in points, as is shown by the enormous increase in completeness, but they are also more accurate than those produced by the model with base-resolution. Finally, the Outdoor sequences are the ones most affected by Super-Resolution, with an overall discrete improvement on all tolerances, but with a noteworthy completeness improvement of 12% for $\tau = 1cm$.

Going to analyze the causes of these behaviors, we must dwell on the nature of the PatchMatch-based algorithm on which this model relies. We remember that the idea of the method is to augment the set of COLMAP’s PatchMatch depth hypotheses with the novel hypothesis based on superpixel concept. This allows the framework to propagate the depth and normal estimates belonging to photometrically stable regions around edges to the superpixel, practically as if it were a matter of estimating the depth of a single big point. In this way, the algorithm manages to effectively reconstruct those regions poor of texture whose depths are typically very difficult to estimate accurately. Keeping this in mind, it is likely that the algorithm is particularly affected by the presence of unwanted artifacts produced by the Super-Resolution and that probably, due to these, it is difficult to choose the correct hypothesis while dealing with piecewise planar priors corresponding to Textureless surfaces. On the other hand, it seems capable of choosing the hypotheses very well when the Super-Resolution scenes to be reconstructed are deep and rich in objects with many details. Given the good overall performance, we can keep the default model’s parameters for the evaluation of the test set.
5.2. TAPA-MVS-SR

Figure 5.5: TAPA-MVS performance over $S_{\text{train}}$ and $S_{\text{train}}^2$. $F1$, accuracy and completeness are computed according with tolerances $\tau = 1, 2, 5, 10, 20, 50$cm. Redline set at 0 represents the basic-resolution model to beat, while the interpolated points represent the difference between the performance of the Super-Resolution model and the aforementioned basic-resolution one, for each metric and for each tolerance.
5.2.2 ETH3D test

We evaluate the performance of the TAPA-MVS model on $S_{test}$ and $S_{test}^2$, leaving its setup unchanged compared to the previous evaluation on $S_{train}$ and $S_{train}^2$.

An analysis of the Overall results shows a different behavior than expected. It is in fact evident from the graphs in Figure 5.7 as from the data in Table 5.7 a general degradation both in terms of completeness and in terms of accuracy, with a consequent collapse of F1 score. Going further, we note, albeit with a slight bias, a behavior similar to the scores obtained in the sequences rich in texture compared to the train set. What we did not expect, and that most influences the general results, is such a different behavior with regard to scenes with regions without textures, where the results from Super-Resolution sequences are characterized by a large collapse of completeness compared to the results obtained previously with the same model.

The reasons for these differences are related to two concepts. The first is the creation of the Super-Resolution algorithm of artifacts in the images produced. The second is the choice of the photometric consistency hypotheses made during the reconstruction phase. In fact, it is likely that improvements in the starting images, as well as prediction errors, are not evident to the human eye, but have a strong impact in the estimation of the depth maps. Some points belonging to areas with homogeneous depth in the base-resolution model are no longer considered as photometrically stable in the Super-Resolution model, and thus they are no longer aggregated by TAPA-MVS in superpixels.

Table 5.7: F1 scores over $S_{test}$ and $S_{test}^2$ with TAPA-MVS. We compare the performance of the Super-Resolution model against those of the basic-resolution one according with the tolerances $\tau$. In this case the Indoor sequences coincide with the Textureless ones, while the Outdoor sequences coincide with the Textured ones.

<table>
<thead>
<tr>
<th>$\tau$ (cm)</th>
<th>Overall</th>
<th>Textureless</th>
<th>Textured</th>
</tr>
</thead>
<tbody>
<tr>
<td>base SR</td>
<td></td>
<td>base SR</td>
<td>base SR</td>
</tr>
<tr>
<td>1</td>
<td>41.97</td>
<td>40.72</td>
<td>37.54</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>58.67</td>
<td>55.07</td>
</tr>
<tr>
<td>5</td>
<td>76.12</td>
<td>73.30</td>
<td>70.08</td>
</tr>
<tr>
<td>10</td>
<td>84.80</td>
<td>83.88</td>
<td>80.13</td>
</tr>
<tr>
<td>20</td>
<td>90.68</td>
<td>91.35</td>
<td>89.42</td>
</tr>
<tr>
<td>50</td>
<td>95.48</td>
<td>95.82</td>
<td>94.80</td>
</tr>
</tbody>
</table>
5.2. TAPA-MVS-SR

![Comparison between two TAPA-MVS depth map sample pairs of Textureless scenes belonging to $S_{\text{test,txl}}$ and $S_{\text{test,txt}}$](image)

From the depth maps samples of the Textureless sequences in Figure 5.6 we can clearly see the effect of the phenomenon just described. We note better estimates than the base-resolution model in areas with more detail, especially in those less uniform areas. What is more evident, however, is the presence of holes in the texture and entire areas belonging to the background of the scene but estimated as very close to the camera. If we want to summarize the reasons for this behavioral difference between train set and test set, it is only possible to say that in the second case there was a combination of artifacts related to Super-Resolution and the choice of PatchMatch hypotheses by TAPA-MVS more unfortunate compared to the first case.

From a qualitative analysis of the scenes reconstructed in 3D, whose images are present in Appendix A, we find coherence with what has been expressed so far. We see an increase in general details in all those outdoor scenes or the more detailed regions of almost all the sequences, especially as details in the regions most distant from the camera. It is also evident that the hypothesis linked to superpixels is often rejected by reconstructing floors and walls in a more fragmented way. However, there is almost always an increase in detail due to the greater presence of information in the input.
Figure 5.7: TAPA-MVS performance over $S_{\text{test}}$ and $S_{\text{test}}^2$. $F_1$, accuracy and completeness are computed according with tolerances $\tau = 1, 2, 5, 10, 20, 50\text{cm}$. Redline set at 0 represents the basic-resolution model to beat, while the interpolated points represent the difference between the performance of the Super-Resolution model and the aforementioned basic-resolution one, for each metric and for each tolerance.
5.3 CasMSVNet-SR

In this section we present the experiments carried out and the results obtained using CasMVSNet, the DeepLearning-based algorithm developed by Gu et al. (2019).

What we expect from this algorithm is to see completely different behaviours compared with the ones demonstrated by COLMAP and TAPA-MVS. Due to its nature, this algorithm does not have the characteristics related to PatchMatch techniques, and therefore we do not foresee the presence of problems in the regions without textures as in the previous cases. We are also curious to observe how a deep neural network manages to exploit Super-Resolution in areas rich in detail.

We will initially observe the behavior of CasMSVNet on $S_{train}$ and $S_{train}^2$ sequences, comparing the each result obtained according to the proposed metrics, and evaluating on the basis of these if it is necessary or not to modify the parameters of the algorithm. Next, we will validate the model on $S_{test}$ and $S_{test}^2$ sequences.

5.3.1 ETH3D train

Like we did for PatchMatch-based pipelines, we reconstruct all the sequences $S_{train}$ and $S_{train}^2$ but this time using CasMVSNet with the parameters setup described in the previous chapter.

From the charts in Figure 5.9 it is possible to say that on average the Super-Resolution impacts positively the DeepLearning-based model. Apparently, this algorithm reacts to a greater presence of pixels than the base-resolution model producing denser point clouds, sometimes such as to include new areas in the reconstruction as shown in Figure 5.8. All this translates into an overall and constant increase in completeness that can be found regardless of the tolerance considered.

In order to better understand the origin of this improvement, we deepen the analysis to the sub-sequences. In fact, if the increase in completeness can be generalized to all the cases, the Textureless or Indoor sequences also enjoy an enhancement in accuracy for low tolerances, unlike Textured or Outdoor ones. It is essential to underline the importance of such an improvement, given that generally, an increase in one of these two metrics with respect to the base-resolution model corresponds to the deterioration of the other one.

From an analysis of the Overall results in the Table 5.8 we observe
Figure 5.8: Qualitative comparison between base-resolution and Super-Resolution reconstructions of electro sequence made with CasMVSNet. In the image on right it is possible to appreciate the presence of novel areas reconstructed.

how Super-Resolution has a very good behaviour on this model, to the point of making it always better than that with base-resolution regardless of the tolerances considered. The same scenario is also found in the Textured, Textureless and Outdoor subsequences, making it clear how much this DeepLearning-based model is able to generalize what has been learned during its training phase and to exploit it in the production of accurate outputs despite the presence of artifacts due to Super-Resolution.

It is also important to remember that, as previously explained, due to computational limitations, the algorithm is forced to downsample the images with Super-Resolution, which however makes them larger and more detailed than those with base-resolution, but that does not allow to fully appreciate the benefits of this approach. However, given the good results obtained, we decide to maintain this setup to evaluate the pipeline on the test set.

Table 5.8: F1 scores over $S_{\text{train}}$ and $S_{\text{train}}^2$ with CasMVSNet. We compare the performance of the Super-Resolution model against those of the basic-resolution one according with the tolerances $\tau$.

<table>
<thead>
<tr>
<th>$\tau$ (cm)</th>
<th>Overall</th>
<th>Indoor</th>
<th>Outdoor</th>
<th>Textureless</th>
<th>Textured</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>base SR</td>
<td>base SR</td>
<td>base SR</td>
<td>base SR</td>
<td>base SR</td>
</tr>
<tr>
<td>1</td>
<td>38.28</td>
<td>39.59</td>
<td>37.24</td>
<td>38.46</td>
<td>38.97</td>
</tr>
<tr>
<td>2</td>
<td>49.00</td>
<td>49.66</td>
<td>48.08</td>
<td>48.09</td>
<td>49.61</td>
</tr>
<tr>
<td>5</td>
<td>60.59</td>
<td>61.15</td>
<td>61.10</td>
<td>60.40</td>
<td>60.25</td>
</tr>
<tr>
<td>10</td>
<td>67.59</td>
<td>68.37</td>
<td>69.39</td>
<td>69.10</td>
<td>66.39</td>
</tr>
<tr>
<td>20</td>
<td>74.01</td>
<td>74.84</td>
<td>77.33</td>
<td>77.25</td>
<td>71.79</td>
</tr>
<tr>
<td>50</td>
<td>82.59</td>
<td>83.61</td>
<td>87.14</td>
<td>87.31</td>
<td>79.56</td>
</tr>
</tbody>
</table>
Figure 5.9: CasMVSNet performance over $S_{train}$ and $S_{train}^2$. F1, accuracy and completeness are computed according with tolerances $\tau = 1, 2, 5, 10, 20, 50cm$. Red line set at 0 represents the basic-resolution model to beat, while the interpolated points represent the difference between the performance of the Super-Resolution model and the aforementioned basic-resolution one, for each metric and for each tolerance.
5.3.2 ETH3D test

We evaluate the performance of the CasMVSNet model on $S_{test}$ and $S_{test}^2$, leaving its setup unchanged compared to the previous evaluation on $S_{train}$ and $S_{train}^2$.

By observing the data in Table 5.9, we have confirmation of what was stated for the train test, and in addition, we can observe a general and consistent improvement in terms of F1 score, which demonstrates us that this model is always subject to improvement if preceded by a Super-Resolution algorithm. Looking at the results of the charts in Figure 5.10, we notice a similar situation compared to the train set, with the exception of the Indoor and Textureless sequences, which in this case undergo a net improvement, albeit slight, for each metric and for each tolerance considered.

From a qualitative point of view, 3D reconstructions starting from Super-Resolution images, shown in the Appendix A of this document, are very dense in the reconstructed areas, are characterized by little noise and, compared to those obtained from base-resolution images are more complete and detailed. More specifically, depending on the reconstructed scene, we note an increase in the details near the camera and an average shallower scene compared to the base-resolution model, or we observe how some details or entire bodies in the vicinity of the camera are ignored and the details in the deeper regions have increased. We are led to believe that both these phenomena are related to the computational limitation and that the full power model can benefit from Super-Resolution in a much more incisive way.

Table 5.9: F1 scores over $S_{test}$ and $S_{test}^2$ with CasMVSNet. We compare the performance of the Super-Resolution model against those of the basic-resolution one according with the tolerances $\tau$. In this case the Indoor sequences coincide with the Textureless ones, while the Outdoor sequences coincide with the Textured ones.

<table>
<thead>
<tr>
<th>$\tau$ (cm)</th>
<th>Overall</th>
<th>Textureless</th>
<th>Textured</th>
</tr>
</thead>
<tbody>
<tr>
<td>base SR</td>
<td>SR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>35.32</td>
<td>35.87</td>
<td>18.44</td>
</tr>
<tr>
<td>2</td>
<td>44.49</td>
<td>45.13</td>
<td>24.93</td>
</tr>
<tr>
<td>5</td>
<td>55.03</td>
<td>56.09</td>
<td>35.19</td>
</tr>
<tr>
<td>10</td>
<td>61.88</td>
<td>63.08</td>
<td>43.72</td>
</tr>
<tr>
<td>20</td>
<td>68.63</td>
<td>69.65</td>
<td>53.70</td>
</tr>
<tr>
<td>50</td>
<td>78.10</td>
<td>79.00</td>
<td>68.51</td>
</tr>
</tbody>
</table>
5.3. CasMSVNet-SR

Figure 5.10: CasMVSNet performance over $S_{test}$ and $S_{test}^2$. F1, accuracy and completeness are computed according with tolerances $\tau = 1, 2, 5, 10, 20, 50$ cm. Redline set at 0 represents the basic-resolution model to beat, while the interpolated points represent the difference between the performance of the Super-Resolution model and the aforementioned basic-resolution one, for each metric and for each tolerance.
5.4 Final Results

In this section, we analyze the behaviour of all three models described above on the entire benchmark, then we compare them and draw the conclusions of this work. We start from an overall comparison, then go to deepen the results by discriminating between scenes rich and poor in texture.

As for COLMAP, we note that it is absolutely the pipeline that has undergone the effects of the Super-Resolution more than its equivalent reconstructed starting from base-resolution images. As can be seen from Figure 5.11, it has in fact achieved a massive improvement in completeness and a not insignificant deterioration in accuracy especially for low tolerances, which on average translate into more detailed but also noisier reconstructions. Comparing it to TAPA-MVS, the other PatchMatch-based algorithm, we notice a different behavior. In fact, given the presence of the hypothesis based on superpixels which made the latter algorithm more performing than the first, we observe a similar but less incisive behavior only for $\tau = 1\text{cm}$, while with increasing tolerance the situation overturns, increasing the accuracy of the reconstructed points but reducing their number. Finally, by combining these two CasMVSNet models, we note that the latter has undergone less intense absolute variations but more homogeneous with respect to tolerances.

As can be observed from Table 5.10, the latter model, as well as COLMAP, apparently benefited from the contribution of Super-Resolution, unlike TAPA-MVS which from a numerical point of view does not differ from its base-resolution model. Before conclusions can be drawn, however, further comparisons are needed.

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
<th>accuracy</th>
<th>completeness</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLMAP</td>
<td>+0.96%</td>
<td>-1.62%</td>
<td>+3.88%</td>
</tr>
<tr>
<td>TAPA-MVS</td>
<td>+0.01%</td>
<td>+0.84%</td>
<td>-0.38%</td>
</tr>
<tr>
<td>CasMVSNet</td>
<td>+0.88%</td>
<td>-1.13%</td>
<td>+1.82%</td>
</tr>
</tbody>
</table>

Table 5.10: Comparison over F1, accuracy and completeness improvements due to Super-Resolution between the proposed Multi-View Stereo pipelines. The scores are computed as unweighted average over the entire ETH3D low-resolution many-view benchmark.
Figure 5.11: COLMAP, TAPA-MVS and CasMVSNet performance over Super-Resolution ETH3D benchmark $S^2$. $F1$, accuracy and completeness are computed according with tolerances $\tau = 1, 2, 5, 10, 20, 50$ cm. For each model, red line set at 0 represents the basic-resolution score to beat, while the interpolated points represent the difference between the performance of the Super-Resolution model and the aforementioned basic-resolution one, for each metric and for each tolerance.
Let’s consider the Textureless sequences. As we can see both from the graphs in Figure 5.12 and from the data in Table 5.11, as previously explained, PatchMatch-based algorithms suffer overall from the presence of artifacts in smooth areas and with few details due to Super-Resolution.

It is however evident that, as far as TAPA-MVS is concerned, the general loss of performance is much greater than COLMAP one, definitive proof of the fact that even slight colour variations of the pixels belonging to these poor areas of texture, carry the algorithm to use different hypotheses for estimating depth maps. We arrive at completely different conclusions in the case of CasMVSNet, which instead seems not to have been penalized by the presence of artifacts, managing to improve overall under each metric used to analyze it. At this point, it is clear that the choice of the MVS algorithm to be improved through Super-Resolution is crucial for this type of scenario to be reconstructed.

It must be remembered that, in the case of the DeepLearning-based algorithm we adopted, the result obtained starting from the base-resolution images results to be incomplete. In fact, the tendency of these families of algorithms is to create a huge number of points, many more than PatchMatch-based techniques, and for this reason, they need much more memory. If, as in this case, there are limitations due to hardware capability, the reconstructed scene will be extremely detailed in the areas closest to the camera and almost completely empty in the more distant ones. What can be seen from the differences with the Super-Resolution counterpart is that the algorithm increases the overall performance, sometimes thanks to an increased number of reconstructed details closer to the camera, sometimes thanks to an increment in the depth of the reconstructed scene.

Table 5.11: Comparison of F1, accuracy and completeness improvements due to Super-Resolution between the proposed Multi-View Stereo pipelines. The scores are computed as unweighted average over the Textureless sequences subset of the ETH3D low-resolution many-view benchmark.

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
<th>accuracy</th>
<th>completeness</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLMAP</td>
<td>+0.13%</td>
<td>-1.27%</td>
<td>+1.66%</td>
</tr>
<tr>
<td>TAPA-MVS</td>
<td>-1.35%</td>
<td>+1.07%</td>
<td>-3.63%</td>
</tr>
<tr>
<td>CasMVSNet</td>
<td>+0.91%</td>
<td>+0.28%</td>
<td>+1.15%</td>
</tr>
</tbody>
</table>
5.4. Final Results

Figure 5.12: COLMAP, TAPA-MVS and CasMVSNet performance over Textureless Super-Resolution ETH3D benchmark $S^2_t$. F1, accuracy and completeness are computed according with tolerances $\tau = 1, 2, 5, 10, 20, 50$cm. For each model, redline set at 0 represents the basic-resolution score to beat, while the interpolated points represent the difference between the performance of the Super-Resolution model and the aforementioned basic-resolution one, for each metric and for each tolerance.
Finally, going to examine the overall results on the Textured sequences, composed of deeper scenes and rich in details than the previous ones, we note how the PatchMatch-based algorithms are able to significantly improve their performance, and this is evident both by observing the graphs in Figure 5.13 that the data in Table 5.12.

As far as COLMAP is concerned, the enormous increase in completeness is paid for with a loss of accuracy, obtaining reconstructions of open spaces and very compact rough surfaces surrounded by noise. As for TAPA-MVS, in this scenario, it is evident that the hypothesis based on superpixels turns out to be a winning card, as the reconstructions of this type of Super-Resolution sequences also increase in completeness, while managing not to lose accuracy as in the case of the previous model. Going to observe the images of the reconstructions in the Appendix, we notice a considerable increase in detail and in some cases even a visible reduction in noise, synonymous with the fact that the algorithm is able to choose the correct hypotheses in this type of scenario.

CasMVSNet reconfirms the conclusions of the analysis of the general case, almost constantly increasing the completeness of the reconstructions regardless of the tolerance considered, with a loss in accuracy compared to the base-resolution model. For the reasons previously explained, also in this case the DeepLearning-based algorithm cannot fully reconstruct the scene regardless of the resolution of the input. However, since the scenes in this sequence are on average deeper than those previously analyzed, it can be seen more clearly that the Super-Resolution allows the model to reconstruct many details of the regions far from the camera in most cases, or focuses on parts of the scene very close to the camera. Either way, the improvements aren’t enough to reconstruct the whole scene due to the hardware limitation.

Table 5.12: Comparison of F1, accuracy and completeness improvements due to Super-Resolution between the proposed Multi-View Stereo pipelines. The scores are computed as unweighted average over the Textured sequences subset of the ETH3D low-resolution many-view benchmark.

<table>
<thead>
<tr>
<th>Model</th>
<th>F1</th>
<th>accuracy</th>
<th>completeness</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLMAP</td>
<td>+1.80%</td>
<td>-1.97%</td>
<td>+6.09%</td>
</tr>
<tr>
<td>TAPA-MVS</td>
<td>+1.36%</td>
<td>+0.61%</td>
<td>+2.86%</td>
</tr>
<tr>
<td>CasMVSNet</td>
<td>+0.85%</td>
<td>-2.55%</td>
<td>+2.48%</td>
</tr>
</tbody>
</table>
Figure 5.13: COLMAP, TAPA-MVS and CasMVSNet performance over Textured Super-Resolution ETH3D benchmark $S_{txd}^2$. F1, accuracy and completeness are computed according with tolerances $\tau = 1, 2, 5, 10, 20, 50$cm. For each model, redline set at 0 represents the basic-resolution score to beat, while the interpolated points represent the difference between the performance of the Super-Resolution model and the aforementioned basic-resolution one, for each metric and for each tolerance.
With reference to Table 5.13 we can conclude that Super-Resolution has as general effect an increase in the number of reconstructed points, regardless of the MVS algorithm used, which translates into an increase in completeness of the reconstructions compared to the base-resolution models. PatchMatch-based algorithms suffer from the presence of artifacts, sometimes producing unexpected effects in regions with little texture, but at the same time enriching with detail all the other areas that instead have a very defined texture. We therefore consider the pipelines based on these algorithms to be a success and recommend their use when the subject of the reconstruction consists of an open environment, with the presence of many objects and rich in rough surfaces. The DeepLearning-based algorithms, on the other hand, react very well to the greater presence of pixels in the input images due to the Super-Resolution, regardless of the type of sequence to be reconstructed. We therefore believe that, whenever possible, this family of algorithms must always be preceded by a Super-Resolution technique to obtain more robust and detailed reconstructions.

Table 5.13: F1, accuracy and completeness scores over ETH3D low-resolution multi-view benchmark. We compare the presented models grouped in many subsets with a tolerance $\tau = 1cm$

<table>
<thead>
<tr>
<th>model</th>
<th>Overall F1</th>
<th>Train F1</th>
<th>Test F1</th>
<th>Overall acc</th>
<th>Train acc</th>
<th>Test acc</th>
<th>Overall compl</th>
<th>Train compl</th>
<th>Test compl</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLMAP (base)</td>
<td>36.58 40.66 33.75</td>
<td>35.8 39.35 33.41</td>
<td>37.35 41.98 34.09</td>
<td>COLMAP (SR)</td>
<td>40.6 37.23 45.77</td>
<td>40 36.97 45.2</td>
<td>41.2 37.49 46.34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAPA-MVS (base)</td>
<td>40.42 41.79 40.33</td>
<td>38.87 42.04 38.22</td>
<td>41.97 41.54 42.44</td>
<td>TAPA-MVS (SR)</td>
<td>41.53 40.66 44.99</td>
<td>42.33 41.08 45.32</td>
<td>40.72 40.23 44.65</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CasMVSNet (base)</td>
<td>36.8 42.37 34.27</td>
<td>38.28 44.74 34.9</td>
<td>35.32 39.99 33.65</td>
<td>CasMVSNet (SR)</td>
<td>37.73 41.38 36.87</td>
<td>39.59 44.59 37.08</td>
<td>35.87 38.16 36.65</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>model</th>
<th>Indoor F1</th>
<th>Outdoor F1</th>
<th>Textured F1</th>
<th>Textureless F1</th>
<th>Indoor acc</th>
<th>Outdoor acc</th>
<th>Textured acc</th>
<th>Textureless acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLMAP (base)</td>
<td>34.39 38.12 31.7</td>
<td>38.04 42.36 35.12</td>
<td>38.94 42.92 36.34</td>
<td>34.21 38.41 31.17</td>
<td>COLMAP (SR)</td>
<td>36.5 35.72 38.2</td>
<td>43.33 38.24 50.82</td>
<td>44.29 38.88 52.41</td>
</tr>
<tr>
<td>TAPA-MVS (base)</td>
<td>41.38 41.56 39.78</td>
<td>41.93 39.539</td>
<td>40.86 40.21 42.2</td>
<td>39.98 43.38 38.46</td>
<td>TAPA-MVS (SR)</td>
<td>37 41.9 34.32</td>
<td>44.54 39.83 52.1</td>
<td>45.07 40.05 53.4</td>
</tr>
<tr>
<td>CasMVSNet (base)</td>
<td>27.84 35.79 23.73</td>
<td>42.77 46.75 41.3</td>
<td>43.21 47.18 42.13</td>
<td>30.38 37.55 26.41</td>
<td>CasMVSNet (SR)</td>
<td>28.92 37.42 24.65</td>
<td>43.61 44.02 45.01</td>
<td>43.63 43.65 45.74</td>
</tr>
</tbody>
</table>
Chapter 6

Conclusions and Future Works

In this thesis, we showed how and under what circumstances it is possible to improve the quality of a 3D reconstruction using Super-Resolution algorithms. To do this, we used many Multi-View Stereo algorithms based on different techniques to reconstruct, evaluate and compare a wide range of scenarios with and without the aid of Super-Resolution. In particular, we used one of the most advanced Single-Image Super-Resolution model, i.e. Deep Back-Projection Network by Haris et al. (2019), placing it on top of Multi-View Stereo pipelines based on both PatchMatch-based, i.e. COLMAP by Schönberger et al. (2016) and TAPA-MVS by Romanoni and Matteucci (2019), and DL-based technologies, i.e. CasMSVNet by Gu et al. (2019), obtaining excellent improvements for both of these families of models in terms of completeness of the reconstructions. We compared the proposed models from various points of view, starting from the depth maps produced up to a quantitative and qualitative analysis of the dense point clouds produced.

Although each algorithm gave a different response to Super-Resolution, we generally found a significant increase in performance when the scenario to be reconstructed consists of an open environment, full of details and the large presence of rough surfaces and objects inside. Conversely, in closed environments and with a large presence of smooth or reflective surfaces such as walls or floors, on average there was an increase in the details at the ends of these poor regions in texture, accompanied however by a lot of noise in the form of points not correctly positioned in space and by holes in the central areas of the aforementioned regions.

As for PatchMatch-based models, we were able to observe an increase
in the depth of the reconstructed scene, while as regards CasMVSNet we witnessed contradictory behaviors. Some scenes were reconstructed with more details near the camera than the base-resolution model, others with greater details in depth. We are sure that these reactions to the Super-Resolution are dictated by the hardware limitation and that having a more powerful calculation capacity, it is possible to obtain not only further improvements from the proposed pipeline, but also equivalent, if not better, reconstructions than those carried out by the PatchMatch-based models.

In conclusion, we recommend the use of Super-Resolution at the head of each Multi-View Stereo algorithm if the object of the reconstruction is an open landscape, since the greater presence of pixels generally leads to a visually and analytically better result, despite the presence of artifacts. As for the scenes of indoor environments or smooth objects, the contribution of Super-Resolution depends a lot on algorithm to algorithm, therefore we recommend experimenting.

This work opens the door to various interesting future projects. As a first step, it is possible to create new hypotheses for the PatchMatch based algorithms that take into account the artifacts generated during the Super-Resolution phase, so as to find a robust solution to all those poor regions in texture. When it is computationally possible, it would be interesting to observe the behavior of a DeepLearning-based algorithm with full potential in the proposed pipeline. Over time it will be possible to replicate the proposed experiments with the use of both newer and more performing Super-Resolution and Multi-View Stereo algorithms, going to study the synergies between the improvements in the individual fields. Furthermore, by exploiting greater hardware capabilities, it is possible to exploit the full potential of the DeepLearning-based pipelines for Multi-View Stereo, which we are sure can lead to even more impressive results, or study the behaviour of these algorithms using greater magnification scales in the Super-Resolution phase. Another interesting idea could be to create an end-to-end autonomous learning architecture able to adapt its weights to obtain the best possible reconstruction. Starting from this, it would be possible to implement both multiple magnification scales, and the most modern approaches based on Generative Adversarial Networks or Graph Neural Networks.

It is in any case certain that, with the advancement of GPU-based
hardware technology and with the growing trend related to autonomous driving, mixed reality and augmented reality, as well as with the great interest that Deep Learning has been experiencing in recent years, these families of algorithms are destined to evolve and we will all have the opportunity to appreciate ever more detailed and wonderful reconstructions.
Bibliography


Appendix A

Reconstructions

In the following pages, we show the reconstructions of the ETH3D benchmark in both the base-resolution and the Super-Resolution versions. For each sequence, we propose the dense point cloud produced by COLMAP by Schönberger et al. (2016), TAPA-MVS Romanoni and Matteucci (2019) and CasMVSNet Gu et al. (2019).
delivery_area

COLMAP

TAPA-MVS

CasMVSNet
COMAP (SR)

$F_1_{avg} : +1.07\%, \quad accuracy_{avg} : -2.56\%, \quad completeness_{avg} : +4.96\%$

TAPA-MVS (SR)

$F_1_{avg} : +2.01\%, \quad accuracy_{avg} : +7.97\%, \quad completeness_{avg} : -5.28\%$

CasMVSNet (SR)

$F_1_{avg} : -1.06\%, \quad accuracy_{avg} : -3.54\%, \quad completeness_{avg} : +0.25\%$
COLMAP (SR)

$F_1_{\text{avg}} : +1.50\%$, $\text{accuracy}_{\text{avg}} : -2.90\%$, $\text{completeness}_{\text{avg}} : +5.50\%$

TAPA-MVS (SR)

$F_1_{\text{avg}} : +1.78\%$, $\text{accuracy}_{\text{avg}} : -5.03\%$, $\text{completeness}_{\text{avg}} : +6.10\%$

CasMVSNet (SR)

$F_1_{\text{avg}} : +1.98\%$, $\text{accuracy}_{\text{avg}} : +0.27\%$, $\text{completeness}_{\text{avg}} : +2.84\%$
Lakeside

COLMAP

TAPA-MVS

CasMVSNet
COLMAP (SR)

\[ F_{1\text{avg}} : +2.23\%, \quad \text{accuracy}_{\text{avg}} : -1.13\%, \quad \text{completeness}_{\text{avg}} : +5.76\% \]

TAPA-MVS (SR)

\[ F_{1\text{avg}} : +2.09\%, \quad \text{accuracy}_{\text{avg}} : +1.43\%, \quad \text{completeness}_{\text{avg}} : +2.81\% \]

CasMVSNet (SR)

\[ F_{1\text{avg}} : +0.24\%, \quad \text{accuracy}_{\text{avg}} : -4.16\%, \quad \text{completeness}_{\text{avg}} : +3.36\% \]
playground

COLMAP

TAPA-MVS

CasMVSNet
COLMAP (SR)

$F_1_{avg} : +1.54\%,\quad accuracy_{avg} : -4.26\%,\quad completeness_{avg} : +7.80\%$

TAPA-MVS (SR)

$F_1_{avg} : +3.12\%,\quad accuracy_{avg} : +5.86\%,\quad completeness_{avg} : +1.13\%$

CasMVSNet (SR)

$F_1_{avg} : +1.38\%,\quad accuracy_{avg} : -5.24\%,\quad completeness_{avg} : +2.03\%$
sand_box

COLMAP

TAPA-MVS

CasMVSNet
COLMAP (SR)

$F_{1\text{avg}} : +0.40\%$,  $accuracy_{\text{avg}} : -3.03\%$,  $completeness_{\text{avg}} : +5.04\%$

TAPA-MVS (SR)

$F_{1\text{avg}} : -2.37\%$,  $accuracy_{\text{avg}} : -6.52\%$,  $completeness_{\text{avg}} : +4.15\%$

CasMVSNet (SR)

$F_{1\text{avg}} : +0.91\%$,  $accuracy_{\text{avg}} : -1.41\%$,  $completeness_{\text{avg}} : +3.02\%$
storage_room

COLMAP

TAPA-MVS

CasMVSNet
storage_room_2

COLMAP

TAPA-MVS

CasMVSNet
COLMAP (SR)

$F_1_{\text{avg}} : -0.92\%$, $\text{accuracy}_{\text{avg}} : -2.11\%$, $\text{completeness}_{\text{avg}} : -0.03\%$

TAPA-MVS (SR)

$F_1_{\text{avg}} : -5.27\%$, $\text{accuracy}_{\text{avg}} : -0.38\%$, $\text{completeness}_{\text{avg}} : -8.32\%$

CasMVSNet (SR)

$F_1_{\text{avg}} : +0.82\%$, $\text{accuracy}_{\text{avg}} : -0.18\%$, $\text{completeness}_{\text{avg}} : +0.84\%$
COLMAP (SR)

$F_1_{\text{avg}} : -1.04\%$, $\text{accuracy}_{\text{avg}} : +1.08\%$, $\text{completeness}_{\text{avg}} : -2.41\%$

TAPA-MVS (SR)

$F_1_{\text{avg}} : -2.7\%$, $\text{accuracy}_{\text{avg}} : -0.16\%$, $\text{completeness}_{\text{avg}} : -4.29\%$

CasMVSNet (SR)

$F_1_{\text{avg}} : +1.17\%$, $\text{accuracy}_{\text{avg}} : +2.72\%$, $\text{completeness}_{\text{avg}} : +0.46\%$
COLMAP (SR)

$F_1_{\text{avg}} : +1.97\%$, $\text{accuracy}_{\text{avg}} : -2.32\%$, $\text{completeness}_{\text{avg}} : +5.16\%$

TAPA-MVS (SR)

$F_1_{\text{avg}} : +1.78\%$, $\text{accuracy}_{\text{avg}} : +0.82\%$, $\text{completeness}_{\text{avg}} : +2.99\%$

CasMVSNet (SR)

$F_1_{\text{avg}} : +0.88\%$, $\text{accuracy}_{\text{avg}} : -0.74\%$, $\text{completeness}_{\text{avg}} : +1.71\%$