Deep Structured Layers for Instance-Level Optimization in 2D and 3D Vision

Filippos Kokkinos

Department of Computer Science

University College London

January 29, 2023

I, Filippos Kokkinos, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

The approach we present in this thesis is that of integrating optimization problems as layers in deep neural networks. Optimization-based modeling provides an additional set of tools enabling the design of powerful neural networks for a wide battery of computer vision tasks. This thesis shows formulations and experiments for vision tasks ranging from image reconstruction to 3D reconstruction.

We first propose an unrolled optimization method with implicit regularization properties for reconstructing images from noisy camera readings. The method resembles an unrolled majorization minimization framework with convolutional neural networks acting as regularizers. We report state-of-the-art performance in image reconstruction on both noisy and noise-free evaluation setups across many datasets.

We further focus on the task of monocular 3D reconstruction of articulated objects using video self-supervision. The proposed method uses a structured layer for accurate object deformation that controls a 3D surface by displacing a small number of learnable handles. While relying on a small set of training data per category for self-supervision, the method obtains state-of-the-art reconstruction accuracy with diverse shapes and viewpoints for multiple articulated objects.

We finally address the shortcomings of the previous method that revolve around regressing the camera pose using multiple hypotheses. We propose a method that recovers a 3D shape from a 2D image by relying solely on 3D-2D correspondences regressed from a convolutional neural network. These correspondences are used in conjunction with an optimization problem to estimate per sample the camera pose and deformation. We quantitatively show the effectiveness of the proposed method on self-supervised 3D reconstruction on multiple categories without the

need for multiple hypotheses.

Impact Statement

We show in this thesis that optimization techniques as components of deep learning networks provide, in return, better accuracy. We experimentally demonstrate that the methodologies presented in this thesis could be adapted to a wide range of vision applications. Beyond these core research problems, there exist multiple real-world use cases that can profit from our advancements.

In detail, the presented unrolled optimization methods for image reconstruction are general enough that could be used for reconstructing images from all sorts of signals like, for example, Magnetic Resonance Imaging (MRI), Global Maxwell Tomography (GMT), or synthetic aperture radar (SAR) data. All imaging systems could benefit from using a mixed optimization and machine learning reconstruction algorithm and compute accurate reconstructions without noise or visual artifacts. This claim was also explored in follow-up work for microscope image reconstruction [1], and its applications can positively impact the medical sciences.

The 3D reconstruction techniques developed as part of this thesis have direct applications in augmented and virtual reality (AR/VR) and the entertainment industry. Modern head-mounted devices like Oculus, Spectacles, and Hololens open fantastic opportunities for real-world impact; however, many technical and research-oriented problems remain unresolved. Techniques presented in this thesis have the potential to enable generic 3D reconstruction of objects, and further improvements could enable the 3D reconstruction of whole scenes from images or videos and their re-projection as holograms in the real world using appropriate optical devices. The CGI industry could adopt the same techniques to speed up movie production and extend the arsenal of techniques currently available for movie making.

To my loving family for their continuous support throughout the years!

Acknowledgements

I have been incredibly fortunate and privileged throughout my entire life to have been given many opportunities that have led me to pursue this thesis research. Over the past four years, this thesis would not have been possible without close collaboration with my two advisors, Iasonas Kokkinos and Stamatios Lefkimmiatis. I am incredibly grateful to both of them for the amount of time and effort they devoted to putting in shape this work and allowing me to grow as a person and academic.

Beyond my advisors, many thanks to all of my close collaborators are in order who have contributed to projects developed during my Ph.D. studies. This includes Francesca Babiloni, Valeriya Pronina, Ioannis Marras, Matteo Maggioni, Gregory Slabaugh, Stefanos Zafeiriou, Dmitry V. Dylov, Marek Kowalski, Stephan Garbin, Virginia Estellers, and Julien Valentin.

This thesis wouldn't have been possible without UCL's excellent research environment, enabling students to pursue their research interests. I want to thank the Computer Science Technical Support Group (TSG), including Ron Gaston, Raymond Ojinnaka, and John Andrews, for their non-stop efforts to keep the CS network and cluster running flawlessly. Many of the presented experiments would not have been possible without their support. I am also very thankful to the administrative staff of UCL, including Gina Baidoo, that helped with the realization of procedures of this thesis, and Antony Hunter, whose research-oriented seminars provided strong guidance for completing the viva. As a UCL student, I also had the unique opportunity to attend Wei Chen's and Treleaven Philip's entrepreneurship seminars. I am also indebted to Abu Mostafa for his kindness, always willing to help the UCL Gower Street building students to the best of his abilities. Without

mentioning the "5.12 lab" doctorate students and post-docs, this list wouldn't have been complete. I am thankful for all the discussions we had with the "5.12 lab" colleagues and not only including Luca Morreale, David Griffiths, Wonbong Jang, Meng Zhang, Changjian Li, Yu-Shiang Wong, Pradyumna Reddy, Philipp Henzler, Eric-Tuan Le, Mohamed Sayed, Edward Bartrum, Giorgos Bouritsas, Ioannis Georgakis, and Athanasios Polymeridis.

While my scientific curiosity was cultivated during my doctorate, the first seeds were planted during my diploma studies at the National Technical University of Athens. I am thankful to Alexandros Potamianos for introducing me to the world of deep learning and helping me through my very first publication. He and his group, including Elias Iosif, Nassos Katsamanis, and Giannis Karamanolakis, have undoubtedly been the stepping stone towards this doctorate.

My Ph.D. studies wouldn't have been complete without my two internships at Huawei's Noah's Ark in London and Microsoft Research in Cambridge. I am thankful for all the conversations and collaborations I had with the researchers in the industry, including Francesca Babiloni, Ioannis Marras, Matteo Maggioni, Gregory Slabaugh, Jiakang Deng, Stefanos Zafeiriou, Steven McDonagh, Ales Leonardis, Ioannis Alexiou, Marek Kowalski, Stephan Garbin, Virginia Estellers, Julien Valentin, Jammie Shotton, Matthew Johnson, Tom Cashman, Andrew Fitzgibbon, Givi Mesihvili, Erroll Wood, Tadas Baltrusaitis, Jingjing Shen, Pashmina Cameron, Martin de la Gorce, Federica Bogo, Vassilis Choutas, and Stan Szymanowicz.

In the end, I want to thank all my friends and family members who supported me with their love and encouragement all these years. Thanks to my parents Elli (Έλλη) and Georgios (Γεώργιος) and my sister Valeria (Βαλέρια) who supported me unconditionally, fueled my curiosity about science and technology and encouraged me to aim higher than what I thought was achievable. I am also grateful to my flatmate and good friend Camilla Yen Hoang Do for shattering the gloominess of lockdowns with her music and songs. Thanks to the Greek community of Moscow, including Evi Manolopoulou, Paraskevi, and Mpampi Vamvaka, for supporting and helping me during my years in Moscow. Finally, I would like to thank

my beloved aunt Natassa (Νατάσσα) who unfortunately is no longer with us. I am eternally indebted to you for always believing in me and supporting me during my undergraduate studies, both financially and practically, even when your health was deteriorating.

Contents

1 Introduction **12**

1.1 3D reconstruction 13

1.1.1 Motivation 13

1.1.2 Single-View Reconstruction 14

1.1.3 Deformation of Surfaces 17

1.2 Structured Layers 19

1.2.1 Structured Layers for Neural Networks 19

1.2.2 Differentiation of Structured layers 20

1.3 Contributions of this thesis 23

1.4 Summary of Publications 25

1.5 Summary of Open-Source Code contributions 26

2 Implicit Regularization for Image Reconstruction **27**

2.1 Introduction 27

2.2 Problem Formulation 31

2.3 Majorization Minimization Method 33

2.4 Residual Denoising Network (ResDNet) 35

2.5 Demosaicking Network Architecture 38

2.6 Network Training 40

2.6.1 Joint Denoising and Demosaicking 40

2.6.2 Training with TBPTT 43

2.7 Experiments 44

2.7.1 Demosaicking Artificial Data 44

2.7.2 Demosaicking Raw Data 44
2.7.3 Demosaicking Non-Bayer CFA 47
2.8 Conclusions 48

3 Self-supervised 3D reconstruction of articulated categories 49
3.1 Introduction 50
3.2 Connection to Related Work 52
3.3 Method Description 53
3.3.1 Learnable Laplacian Solver 54
3.3.2 Motion-based 3D supervision 58
3.3.3 Optimization-based learning and refinement 60
3.4 Experimental Results 61
3.4.1 Model architecture 61
3.4.2 Per-sample optimization training framework 62
3.4.3 Data 63
3.4.4 Results 65
3.4.5 Failure Cases 71
3.5 Conclusions 72

4 Correspondence-driven monocular 3D category reconstruction 73
4.1 Introduction 73
4.2 Related Work 76
4.3 To-The-Point Monocular 3D Mesh Reconstruction 77
4.3.1 Predicting 3D to 2D Correspondences 78
4.3.2 Estimation of Pose and Deformation 78
4.3.3 Texture 82
4.3.4 Loss terms 82
4.4 Experiments 85
4.4.1 Datasets and Metrics 85
4.4.2 Quantitative Results 86
4.4.3 Computational Analysis 88

4.4.4 Failure Cases 91
4.5 Discussion 92

5 Concluding Remarks and Future Directions **96**
5.1 Summary of contributions 97
5.2 Future Directions 99

Bibliography **103**

List of Figures

1.1 Sample results of this thesis. 13

2.1 **Illustration of the residual denoiser**; a core module of our framework. The deep learning based denoiser is used in every iteration to provide an approximate solution of the surrogate function. 35

2.2 **A graphical representation** of the proposed iterative neural network. We have omitted the extrapolation steps for clarity. 38

2.3 **Increasing the number of iterations**, our deep network is capable to achieve increasingly better reconstruction with the same number of parameters. Also, it can be seen that given a large number of iterations, the method is capable to achieve virtually the same performance with and without proper initialization. 42

2.4 **Results** comparing the depth of the denoiser versus the number of iteration on the noisy MSR Dataset. The heat-map depicts the ability of our method to generalize by decreasing the number of parameters and simultaneously increasing the number of iterations. 42

2.5 **Comparison** of our network with other competing techniques on images from the noisy MSR Dataset. From these results it is clear that our method is capable of removing the noise while keeping fine details. On the contrary, the rest of the methods either fail to denoise or they oversmooth the images. 46

3.1 **Quadruped reconstructions** of our proposed method. We provide renderings of the 3D reconstruction using the estimated camera pose, a different viewpoint and the texture reconstruction. 67

3.2 **Handle Influence:** We visualize each row of the matrix **D** for the horse class. We observe that the associations between the learned handles and the 3D points are localized to areas of intuitive interest such as legs, tail and the head which allow for accurate deformation of the template. 68

3.3 **Bird Reconstructions:** Qualitative comparisons between our method, CMR and ACSM with images from the CUB test set. 69

3.4 **Visualization of the predicted deformations** for several objects by depicting the mean shape in the center and the first 3 modes obtained by PCA on the handle estimates obtained across the dataset. 70

3.5 Reconstruction comparisons of our method and ACSM [2] for the object horse. 71

3.6 **Failure Cases:** We visualize some failure modes of our method. The columns present the input image, 3D reconstruction from the predicted viewpoint and a different one and the predicted texture . . 72

4.1 **Overview of our method:** Given an image we use a network ϕ_θ to regress the 2D positions **u** corresponding to the 3D vertices of a template; we then use a differentiable optimization method to compute the rigid (camera) and non-rigid (mesh) pose: in every iteration we refine our camera and mesh pose estimate to minimize the reprojection error between **u** and the reprojected mesh (visualized on top of the input image). The end result is the monocular 3D reconstruction of the observed object, comprising the object's deformed shape, camera pose and texture. 78

4.2 **Bird reconstructions** For each input image we provide the results of CMR [3] and UCMR [2] alongside with our method. We visualize the input image, predictions from prior works and TTP's predicted correspondence (**u**), mesh reconstruction and textured mesh from two viewpoints. We observe that we better capture texture details, deformation and pose estimation. 90

4.3 **Pascal3D+ results** We show predictions of our TTP method for test set images. For each input image we visualize the 3D shape from the predicted camera, the textured shape and a new view. 92

4.4 **Failure Cases:** We visualize some failure modes of our method. The columns present the input image, the predicted 2D points, and 3D reconstruction with and without texture. 93

4.5 **Basis Visualization:** We visualize 8 basis components $T + B_i$ of a trained TTP experiment for the CUB dataset. For each component we visualize a side and a top view. 94

4.6 **Comparison of TTP trained with and without keypoint supervision**. We observe that TTP performs equally well on camera pose estimation with and without keypoint supervision. However, keypoint supervision allows for more accurate mesh deformation. . . . 95

List of Tables

2.1 **Quantitative Comparison** of our system to state-of-the-art techniques on the demosaick-only scenario on artificial data in terms of Peak signal-to-noise ratio (PSNR) performance. 45

2.2 **PSNR performance** of different methods in both linear and sRGB spaces. The results of methods that cannot perform denoising are not included for the noisy scenario. The color space in the brackets indicates the particular color space of the employed training dataset. We exclude PSNR scores of methods who are pure demosaicking from the noisy columns. 47

2.3 **Evaluation** on noise-free linear data with the non-Bayer mosaick pattern Fuji XTrans. 47

3.1 **Ablation of deformation layer on CUB:** Even with only 8 points, our handle-based approach outperforms all competing methods in terms of PCK, while with more handles both the mIoU and PCK scores improve further. 61

3.2 **Keypoint Reprojection Accuracy** We report PCK accuracy (higher is better) achieved by the methods [4, 2] for three different objects. We also indicate datasets used to train each method alongside with their source of supervision. 64

3.3 **Ablation of deformation layer on CUB:** Even when using only 8 points, our handle-based approach outperforms all competing methods in terms of PCK, while with more handles both the mIoU and PCK scores further improve. 65

3.4 **Ablation of motion and optimization** based reconstruction for horses and tigers classes. 65

4.1 **Evaluation** of TTP performance on the CUB [5] dataset. We report mean and standard deviation (in parantheses, where applicable) of 2D mIoU and keypoint re-projection accuracy (PCK) along with related supervision signals for recent monocular 3D reconstruction methods. 86

4.2 **Performance** of TTP method through iterations for pose and deformation estimation. We achieve the best results with more iterations, but even a single iteration suffices for competitive scores. 86

4.3 **Ablation on losses**. 88

4.4 **Ablation on number of basis components**. 89

4.5 **PASCAL3D+ evaluation**. We provide numerical score of TTP with and without keypoint supervision during training. We observe that *even without keypoint supervision* TTP is competitive with the other methods which, except for UCMR, require keypoints. 89

4.6 **Run time analysis** in milliseconds of various self-supervised 3D methods. All benchmarks were run 20 times using images of size 256x256 and we report the average run times. 90

Chapter 1

Introduction

This thesis aims to embed prior knowledge in Convolutional Neural Networks (CNNs) for 3D and image reconstruction problems of computer vision. Starting from an inverse imaging work with a learnable optimization-based approach, we show that the per-sample estimation of reconstructed images provides more accurate and robust reconstructions than directly regressing images with neural networks. We then focus on this thesis's main contribution, which is the integration of optimization problems as layers in neural networks to accurately estimate 3D reconstructions in the form of deformed 3D shapes and camera poses in weakly supervised setups. We refer to layers implementing optimization problems as part of bigger neural networks as structured layers. Our primary contributions in this thesis for optimization-based modeling embedded in CNNs in the form of layers are the following:

- A weakly-supervised approach for 3D reconstruction with CNNs using video correspondences and a robust per sample estimation of deformation
- An extension of our approach to robustly estimate both camera pose and deformation per sample from unstructured image collections
- A per sample optimization approach with CNN regularizers for inverse imaging problems

To position our contributions with respect to the broader 3D reconstruction context, in Section 1.1 we begin by first defining 3D reconstruction in the context

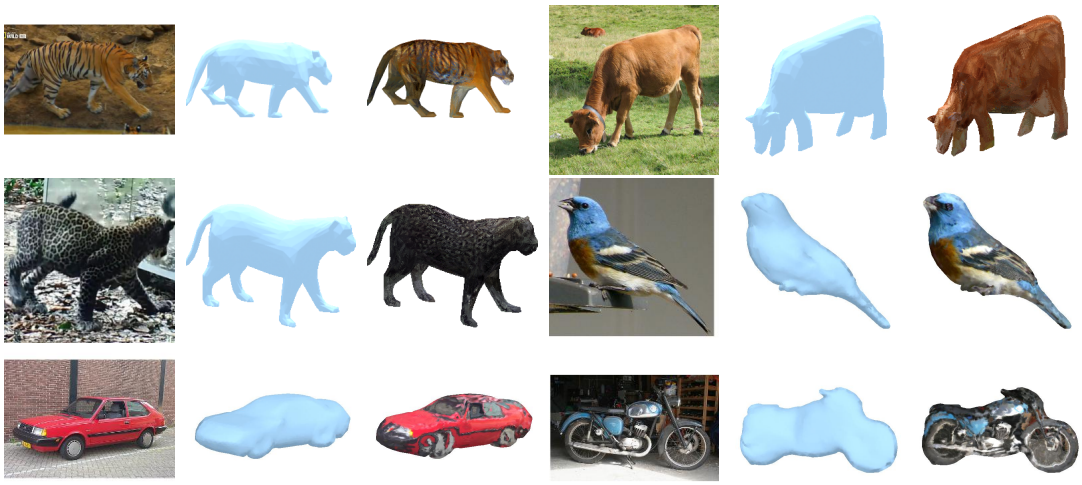


Figure 1.1: Sample results of this thesis.

of this thesis, followed by a review of the relevant 3D deep learning literature. We then give an overview of structured layers in Section 1.2, the challenges they exhibit when embedded in neural networks, and their applications in the general deep learning literature. Finally, in Section 1.3 we summarize the contributions of this thesis. In order to keep the presentation contained we present the literature of image reconstruction only on Chapter 2.

1.1 3D reconstruction

1.1.1 Motivation

In the context of this thesis, we use the term 3D reconstruction to describe computer vision problems where the objective is to reconstruct an object in terms of a 3D dense surface, texture and camera pose. Although there has been significant progress in 3D reconstruction, most of the research assumes the availability of multiple views and high-quality 3D datasets, while much less attention has been given to the ill-posed problem of monocular 3D reconstruction. Humans have the ability to accurately guess the structure of an object depicted in a single image from any viewpoint. Even for non-rigid objects like animals, a person given a single image can easily decompose the depicted object to its 3D shape, its articulated parts and even the viewpoint used to capture the photograph. Modern solutions for this

challenging problem are limited, especially in the case of arbitrary high articulated objects. Hence this thesis aims to develop methods for single-view 3D reconstruction of non-rigid objects where the object category may be known, but no other information such as camera calibration, scene, depth or lighting is known.

1.1.2 Single-View Reconstruction

There are several methods for single-view reconstruction which can be grouped into three different categories, model-based, Shape from X and 3D from image collections.

1.1.2.1 Model-based Reconstruction

Model-based methods rely on the existence of accurate parametric 3D models learned, referred to as 3D Morphable Model (3DMM), using high-quality 3D scans. All model-based methods attempt to fit the parametric 3D model to a single image for single-view reconstruction. The seminal work of Blanz and Vetter [6] built a high-resolution morphable model of a 3D textured face mesh from 3D scans. Subsequently, the method fits the model using non-linear optimization to a single image to reconstruct the shape and albedo. The algorithm solves for the 3D shape and texture parameters that minimize the differences between the rendered model and the image constrained from linear statistical models of facial texture and shape. For many years 3DMMs and their variants were the method of choice for 3D face reconstruction [7, 8, 9] and can still perform comparably to the state-of-the-art in unconstrained 3D shape estimation [10]. Hassner and Basri [11] recover a 3D mesh from a depth estimate of segmented faces and humans. Approaches working with extensive collections of images work on top of keypoint detectors [12] which allow fitting in a fully automated manner [13, 14, 15]. Recently, the focus has shifted towards harnessing CNNs for 3D shape and texture reconstruction with either regressing the parameters of 3DMMs based on an input image [16] or using 3DMM to synthesize an image [17, 18]. The rendering neural network proposed in [19] learns facial deformations and view-specific textures using a variational autoencoder. Gecer et al.[20] proposed an extension where 3DMMs are fitted using a generative adversar-

ial network [21] (GAN) and a differentiable renderer that reconstructs faces with photorealistic quality under arbitrary recording conditions.

Beyond faces, model-based approaches have been proposed for 3D reconstructing humans from a single image. Early works for human reconstruction fit primitive geometric shapes related by a kinematic skeleton to silhouettes [22, 23]. In Hasler et al. [24] the SCAPE [25] parametric body model is fitted to silhouettes using known segmentations and 3D to 2D correspondences. Kulkarni et al. [26] estimates body pose from single images using an articulated 3D human model together with a probabilistic framework. A major leap in performance came with the introduction of SMPL [27] which, unlike SCAPE, has explicit 3D joints that are related to the 3D surface of the body, enabling inference of shape from joints. SMPL is also a linear statistical model where the coefficients can be easily estimated using optimization [28, 29, 30] or regressed using deep neural networks [31, 32]. Using an off-the-shelf keypoint detector, Bogo et al. proposed SMPLify [28] which fits SMPL to 2D keypoints using strong priors to guide the optimization. The recent method SPIN [33] regresses the parameters of SMPL using a CNN and subsequently uses them as initialization for an iterative fitting routine that aligns the model to the 2D keypoints. On the other end of the spectrum, many methods [34, 30, 35] rely exclusively on regression to address the problem of human 3D reconstruction. Kanazawa et al. [35] propose an end-to-end framework for the reconstruction of 3D meshes of humans from a single image via minimization of reprojection losses of keypoints. This training strategy allows the model to be trained using images in the wild that only have ground truth 2D annotations.

There is little work that addresses the modeling challenges of animals. The difficulty of handling live animals and the range of sizes, shapes, and intraclass variability make traditional scanning techniques inapplicable. SMAL models both articulations and intra-class variation well for a broad category of classes, including lions, cats, tigers, dogs, horses, and hippos. SMAL [36], inspired by the SMPL model, learns a parametric model from a small set of 3D scans of toy figurines in arbitrary poses. The same parametric model is used in [37] to predict the 3D dense

surface and camera pose of zebras in the wild using a single image as input. In detail, a deep neural network is trained to regress SMAL parameters, having been trained on an extensive collection of synthetic images of zebras. Biggs et al. [38] propose an iterative fitting routine combined with SMAL as a detailed 3D prior to 3D reconstruct dogs from monocular internet images. In Chapters 3 and 4, we propose two methods to 3D reconstruct animals without any 3D parametric model or 3D scanning, which allow us to support animals that the parametric models don't support.

1.1.2.2 Shape from X

Shape from Shading [39, 40] is one of the earliest single-view 3D reconstruction methods that compute the three-dimensional surface from one image of that surface. Following the success of Shape from Shading, other derivatives were proposed like Shape from Texture [41], Shape from Defocus [42, 43] and Shape from Specularities [44]. The recent work of Barron and Malik [45] unified multiple Shape from X approaches by solving for the shape, reflectance, and illumination using a single image of an object and its' foreground mask. The proposed optimization-based solution searches for the most likely explanation of a single image using a set of appearance and illumination priors. Shape from Silhouette [46, 47] recovers 3D models from a single silhouette image of an object using interactive contour labeling [48, 49]. Vicente and Agapito [50] built on these approaches to 3D reconstruct animals from a reference silhouette and a small set of user-annotated keypoints.

A separate line of work solves single-view reconstruction by deforming a 3D template to match the surface depicted in a single image. Shape from Template [51, 52] focuses on reconstructing inextensible surfaces such as a piece of paper where the area of the surface remains constant under deformations.

1.1.2.3 3D from Image Collections

A new challenge in computer vision is the reconstruction of a target object from a single image, using an image collection of similar objects [53, 54, 4, 55]. Given multiple instances of an object category, the idea is that all different object instances share a similar 3D shape which can be used as surrogate viewpoints to apply Struc-

ture from Motion (SfM) techniques. The early work of [56] demonstrates the applicability of the idea but relies on ground-truth part segmentations to establish correspondences between all images. Kar et al. [54] learns a deformable category-specific 3D shape using foreground masks and camera parameters estimated using SfM. Carreira et al. [53] establishes dense correspondences between all pairs of training images. A target image is then matched to training images of similar viewpoints, thus establishing correspondences to the rest of the training data. These correspondences are combined with an SfM technique to obtain a 3D reconstruction. Cashman et al. [57] learn a morphable model of dolphin shapes from 2D images using a few keypoints and silhouettes to bootstrap the reconstruction process. Reinert et al. [58] extract semi-automatically a 3D model from video sequences using an interactive sketching and tracking approach. The 3D shape is obtained by fitting and tracking cylindrical primitives over multiple frames.

Recent efforts have focused on achieving 3D reconstruction with minimal supervision by removing the need for manual annotation either part segmentation or camera poses; instead, the focus is shifted towards self-supervised correspondence estimation [3, 59, 60, 61]. The common ground of correspondence-based loss is that if the 3D shape is predicted accurately, it should consistently project to the image with the 2D observations in a pixel-by-pixel sense. The geometric cycle loss terms of [62, 4, 2] are explicitly phrased in terms of correspondences established from UV maps. 3D to 2D cycles can also be defined on texture [3, 62, 61, 63, 60] to penalize the differences of the reconstructed texture and the colored image observation.

In Chapter 3 and 4 we propose methods that do not require any annotations for camera pose or part segmentations at training and test time.

1.1.3 Deformation of Surfaces

Deformation of 3D shapes is a ubiquitous task, arising in many vision and graphics applications. Mesh deformation techniques take an input 3D dense surface, usually a mesh, and deform it to fit 3D positional constraints and minimize the surface distortion. Linear deformations are commonly used for deforming models of faces [6], people [27] and animals [36] due to their simplicity and easy adoption as a compo-

nent in learnable 3D reconstruction pipelines [28, 33, 36]. The main drawback of linear deformations is the high-quality 3D scans needed to learn a linear basis that deforms meshes accurately.

Applications in movie or video game productions require deformable meshes with high fidelity, so artists commonly model them. Deforming meshes for thousands of frames requires many person-hours, so recent works have focused on interpolating and extrapolating deformation from a small number of exemplar meshes. The exemplar meshes provided by the artists are subsequently used to learn deformations that lie in the space of exemplars' deformations [64, 65, 66]. Bailey et al. [67] present a method that reduces the time required to compute mesh deformation for films using neural networks, a skeleton rig and a few exemplar meshes.

For free-form deformations where the positional constraints are displacements of several 3D vertices, deformation is achieved with minimization of the elastic energy [46], local rigidity constraints [68] or preservation of local differential properties [69]. Free-form deformations and their various constraints are better suited for monocular 3D reconstruction tasks [61, 2] due to the lack of proper 3D supervision. In Chapter 3 we present a method for learnable deformations that preserves the surfaces of a 3D mesh while adhering to positional constraints provided by a neural network. Similarly to free-form deformations, cage-based deformations [70, 71, 72, 73] enclose a shape with a coarse cage mesh, and all surface points are written as linear combinations of the cage vertices. Recently, Yifan et al. [74] propose a learnable representation for shape deformation using cages and a deep network predicting deformations by controlling the cage.

In the next section, we introduce essential techniques needed to develop structured layers for 3D reconstruction tasks like those presented in Chapter 3 and 4 and ways to back-propagate through them.

1.2 Structured Layers

1.2.1 Structured Layers for Neural Networks

Modern deep learning models are composed of parametrized layers organized in a directed graph with a feed-forward structure. Each processing node implements a function that transforms the node's input to an output. End-to-end learning is then achieved by back-propagating an error signal as a gradient of some global objective back to all the processing nodes and adjusting the parameters to minimize the objective. Back-propagation is achieved with automatic differentiation that computes the derivatives of the output of a processing node with respect to the input and the chain rule of differentiation. At the same time, back-propagation constraints modern deep learning models to solely comprise processing nodes computing continuous numerical representations with well-behaved gradients. Non-differentiable functions where the gradient or the sub-gradient is undefined are not suitable for modern deep learning models like the operations argmin and argmax commonly found in optimization.

This thesis proposes solutions that use classic constrained and unconstrained optimization algorithms as a modular component of larger, end-to-end learnable networks. These modular components are called Structured Layers because they impose structural dependencies in the output in the form of error minimization that purely feed-forward models like CNNs cannot explicitly capture. Energy-based learning methods depend on an energy function $E_\theta(x,y) : \mathbb{X} \times \mathbb{Y} \rightarrow \mathbb{R}$ parametrized with θ that measures the fit between an input x and an output y. The energy function mathematically captures important domain knowledge in the output space of a machine learning task. The solution $\hat{y}$ is inferred as the stationary point of an optimization problem

$$\hat{y} = \underset{y}{\operatorname{argmin}} E_\theta(x,y). \tag{1.1}$$

This formulation is powerful for modelling and learning purposes and subsumes many modern machine learning methods like the deep feed-forward models. Energy-based methods have been used for over two decades, and we refer the reader

to this tutorial [75] for an extensive description of the methods. The domain knowledge captured from an energy function E_θ depends on its formulation. This thesis extends the concept of a generic neural network layer to include camera pose and constrained surface deformations solvers and bi-level optimization for image reconstruction. The proposed solutions use effectively the non-linear underlying physical and mathematical models known to us apriori rather than having to re-learn these within the network. Furthermore, structured components provide guarantees and enforce hard constraints on representations within a model, such as predicting valid rotations belonging to euclidean groups or surface deformations that maintain surface details intact.

1.2.2 Differentiation of Structured layers

The ability to differentiate through structured layers relies on the complexity of the optimization task that is solved. Iteration unrolling is a popular choice for unconstrained optimization problems where a certain number of iterations is unrolled and the back-propagated gradients are computed with the chain rule. One the other hand, if an optimization problem contains non-differentiable operations or constraints differentiation is achieved with the implicit function theorem.

1.2.2.1 Iteration Unrolling

The solution to the minimization problem of Equation (1.1) often cannot be computed analytically and in a closed-form. A viable alternative is unrolling first-order gradient iterations of Equation (1.1) as an approximation. The optimization problem will commonly have the form $x^* = \underset{x}{\operatorname{argmin}} f_\theta(x)$, where the solution is obtained with the gradient descent update rule $x_{i+1} = x_i - \alpha \nabla_x f_\theta(x)$ given an initial guess x_0. The estimate x_N of the last step N is used as the output for loss and back-propagation computation. The number of steps N required to have a good approximate solution depends on the convergence rate of the gradient descent for the energy function E_θ. Simultaneously, the maximum number of unrolled iterations is bounded from the available memory since all intermediate computations are stored for gradient computation. In Chapter 2, we provide a framework for Iterative Neural Net-

works (INN), which unroll iterations of optimization schemes for a theoretically infinite number of iterations without any memory constraints. We apply the proposed method to image restoration tasks successfully over standard feed-forward neural networks.

Back-propagation through unrolled optimization is achieved using automatic differentiation and is widely supported by all major deep learning frameworks such as Tensorflow [76] and Pytorch [77]. This technique has been applied to many tasks across different machine learning and computer vision fields. We highlight some key works using unrolled optimization. The earliest work is [78] which proposes back-propagation strategies for a fixed number of iterations of gradient descent or LBFGS [79]. The authors of [80] propose a structured model that combines Markov random fields with deep learning, and it is trained using unrolled optimization.

Schmidt et al. [81] proposed an iterative unrolled technique for image denoising and deblurring, combining random fields with an optimization algorithm as a single unit. The seminal work of [81] inspired many other follow up works [82, 83, 84, 1, 85, 86, 87] that explored the applicability of the method in various reconstruction tasks, including microscopy and MRI reconstructions. In many recent works [84, 1, 85, 86, 87], the authors use CNNs instead of random fields to learn priors about the task at hand, which are trained via automatic differentiation of the unrolled iterations.

Beyond low-level vision, unrolled structured layers are found in dense labelling tasks that assign one label to each pixel of an image. Most dense labelling methods capture interdependencies in the output space of neural networks to couple local and global contexts before assigning a label for each pixel or image patch. The authors of [88, 89, 90, 91] replace the last layer of convolutional neural networks with conditional random fields for semantic segmentation. These end-to-end trainable networks contain optimization as part of their architecture to perform mean-field inference for a fixed amount of iteration. Differentiation through the mean-field inference is performed automatically through the unrolled iteration. More recently, Chandra et al. [92, 93] provide closed-form expressions for the gradients

needed for inference, thus eliminating the need for a predefined number of inference iterations.

Beyond 2D vision, structure layers can be found in 3D reconstructions tasks. A fully differentiable dense SLAM solver is presented in [94] to estimate maps and trajectories from colour images. The method unrolls iterations of a Levenberg Marquardt solver to estimate the camera pose and map fusion, enabling back-propagation from 3D maps to 2D pixels. The authors of [33] propose a method with parametric model-fitting in a deep neural network for 3D human reconstruction that is end-to-end differentiable. In detail, an initial estimate is regressed from a neural network to initialize an unrolled optimization framework that fits a parametric model of the human body to 2D observations.

1.2.2.2 Implicit Differentiation

If a closed-form solution to the minimization problem of Equation (1.1) is available, then the gradient can be computed analytically. Common optimization formulations with closed-form solutions are the quadratic [81] or Tikhonov regularized problems [1] whose gradients end up being the solution of a linear system. However, there can be optimization objectives where differentiation through the argmin operator of Equation (1.1) cannot be computed analytically with explicit methods or iteratively with unrolled optimization. In these cases, differentiation is achieved with bilevel optimization using the implicit function theorem. Implicit function analysis [95] focuses on solving an equation $f(p,x) = 0$ for x as a function s of p, i.e. $x = s(p)$. Implicit differentiation considers how to differentiate the solution mapping with respect to the parameters, i.e. $\nabla_p s(p)$ and it is presented in Dontchev and Rockafellar [95] as follows.

Theorem 1 *(Implicit function theorem). Let $f : \mathbb{R}^d \times \mathbb{R}^n \to \mathbb{R}^n$ be continuously differentiable in a neighborhood of $(\bar{p},\bar{x})$ and such that $f(\bar{p},\bar{x}) = 0$, and let the partial Jacobian of f with respect to x at $(\bar{p},\bar{x})$, namely $\nabla_x f(\bar{p},\bar{x})$, be nonsingular. Then the solution mapping $S(p) = \{x \in \mathbb{R}^n \mid f(p,x) = 0\}$ has a single-valued localization s around $\bar{p}$ for $\bar{x}$ which is continuously differentiable in a neighborhood Q of $\bar{p}$ with Jacobian satisfying $\nabla s(p) = -\nabla_x f(p,s(p))^{-1} \nabla_p f(p,s(p))$ for every $p \in Q$.*

The implicit function theorem has been used in conjunction with the level-set method that was developed in 1979 by Alain Dervieux [**?**] and subsequently popularized by Stanley Osher [**?**]. A seminal applications of implicit function theorem in computer vision is the method proposed by Mairal et al. [96] where they differentiate the LASSO problem to learn supervised dictionaries for image restoration and classification. Gould et al. [97] propose a class of models where gradients are computed based on the desired behaviour using the implicit function theorem rather than an explicit forward function which allows differentiation even in the case where non-differentiable operations are used. In a follow-up, work [98] estimate the position and orientation of a camera from a set of 2D image pixels and 3D points without prior knowledge of correspondences using geometric optimization. The geometric optimization runs RANSAC [99] to drop outliers which is non-differentiable; however, gradient estimation through this layer is achieved with the implicit function theorem. In the context of video classification, Fernando and Gould [100, 101] show how to differentiate through a rank-pooling operator [102] within a deep learning model, which involves solving a support vector regression problem. Lee et al. [103] consider the problem of few-shot learning for visual recognition. A deep network learns linear classifiers as a basis for generalizing to new classes. Training happens as part of a neural network layer, and it is achieved with quadratic programming (QP) solver. Differentiation through the QP solver is achieved with bi-level optimization in the form of implicit function differentiation.

1.3 Contributions of this thesis

Having described the most relevant literature on 3D reconstruction and structured layers in deep learning, we now give an overview of our contributions in this thesis. The following chapters will discuss each of these in detail.

Unrolled Optimization Schemes (Chapter 2) Building on majorization-minimization framework we develop an unrolled optimization method for reconstructing images from camera readings. We derive the analytic expressions of each iteration and use neural networks as regularizers fitted on available training data.

The approach has a transparent interpretation as a regularization technique compared to black-box approaches mapping inputs to outputs. The method is end-to-end differentiable however training unrolled optimization schemes exhibit significant memory overhead. We present a modified version of the truncated backpropagation through time to circumvent this constraint. We demonstrate the utility of our approach across many datasets, showing substantial improvements over strong black-box baselines.

Structured layers for pose and deformation estimation (Chapter 3, 4) Building on structured layers, we focus on implicit differentiation which allow us to introduce optimization as layers in neural networks. Implicit differentiation is used in this thesis to solve pose estimation and deformation problems relevant to 3D reconstruction.

In Chapter 3, we propose a method that uses self-supervision to reconstruct in 3D highly non-rigid objects from images and videos. We introduce an interpretable model of 3D template deformations that controls a 3D surface through the displacement of a small number of learnable anchor points. This structured layer relies on Laplacian regularization and enables backpropagation through implicit differentiation. We also employ a per-sample numerical optimization approach that jointly optimizes over mesh displacements and cameras, boosting accuracy as test time post-processing. We demonstrate the method's efficacy with state-of-the-art reconstructions for multiple articulated object categories.

In Chapter 4 we simplify the 3D reconstruction method of Chapter 3 and compute both the camera pose and deformation using structured layers. Overcoming the technical hurdles of the previous approach, we 3D reconstruct objects without the need for multiple hypotheses and strong regularization. Instead, we simplify the network to a single CNN that predicts 2D-3D correspondences between an input image and a generic category template. By relying on these 2D-3D correspondences, we use a structured layer to replace CNN-based regression of camera pose and non-rigid deformation. The whole system is end-to-end differentiable, while backpropagation through the structured layer is enabled using implicit differentiation.

We demonstrate systematic improvements on multiple categories and substantial improvements over multi-hypothesis approaches.

1.4 Summary of Publications

The contents of Section 2 appear in:

- [104] Kokkinos, Filippos, and Stamatios Lefkimmiatis. "Deep image demosaicking using a cascade of convolutional residual denoising networks." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.
- [105] Kokkinos, Filippos, and Stamatios Lefkimmiatis. "Iterative residual cnns for burst photography applications." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019.
- [55] Kokkinos, Filippos, and Stamatios Lefkimmiatis. "Iterative joint image demosaicking and denoising using a residual denoising network." *IEEE Transactions on Image Processing* (2019): 4177-4188.

The contents of Section 3 appear in:

- [61] Kokkinos, Filippos, and Iasonas Kokkinos. "Learning monocular 3D reconstruction of articulated categories from motion." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2021.

The contents of Section 4 appears in:

- [106] Kokkinos, Filippos, and Iasonas Kokkinos. "To The Point: Correspondence-driven monocular 3D category reconstruction." *Conference on Neural Information Processing Systems* (2021).

Contribution to research articles as a non-primary author:

- [1] Valeriya Pronina, Filippos Kokkinos, Dmitry V. Dylov, Stamatios Lefkimmiatis "Microscopy Image Restoration with Deep Wiener-Kolmogorov filters." *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020.

- [107] Francesca Babiloni, Ioannis Marras, Filippos Kokkinos, Jiankang Deng, Grigorios Chrysos, Stefanos Zafeiriou "Poly-NL: Linear Complexity Non-local Layers with Polynomials." *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (2021).

Pre-prints:

- [108] Filippos Kokkinos, Ioannis Marras, Matteo Maggioni, Gregory Slabaugh, Stefanos Zafeiriou "Pixel Adaptive Filtering Units." *arXiv preprint arXiv:1911.10581* (2019).

1.5 Summary of Open-Source Code contributions

The code and experiments developed for this thesis are open and publicly available online:

- https://github.com/fkokkinos/deep_demosaick: Pytorch code for the differentiable Majorization Minimization experiments for image reconstruction
- https://github.com/fkokkinos/deep_burst: Pytorch code for the differentiable Proximal Gradient Descent experiments and their application to burst denoising and demosaicking
- https://github.com/fkokkinos/acfm_video_3d_reconstruction: Pytorch code for the articulated mesh reconstruction of animals from monocular images or videos
- https://fkokkinos.github.io/to_the_point/: Pytorch code for the implicit camera optimization problem and their application to self-supervised monocular 3D reconstruction.

Beyond the dissertation projects, I have contributed to the project:

- https://github.com/vpronina/DeepWienerRestoration: Pytorch code for microscopy image reconstruction and deblurring using differentiable quadratic solvers

Chapter 2

Implicit Regularization for Image Reconstruction

In this chapter, we present an optimization unrolling method with implicit regularization properties for the reconstruction of images from camera readings. While there are several machine learning systems that have been recently introduced to solve this problem, we propose a novel algorithm which is inspired by classical image regularization methods, large-scale optimization and deep learning techniques. The method is derived from first principles and unrolls a learnable majorization minimization framework which is fitted to the available training data using standard back-propagation practices. Consequently, our derived neural network has a transparent and clear interpretation compared to other black-box data driven approaches. Our extensive experimentation line demonstrates that the proposed network outperforms any previous approaches on both noisy and noise-free data across many different datasets.

The contents of this chapter were presented to the European Conference on Computer Vision (ECCV) and an extension of the presented work was published on the IEEE Transactions on Image Processing.

2.1 Introduction

Traditionally, high resolution images from a digital camera are the end result of a processing pipeline that transforms light intensity readings to images. The image

processing pipeline is typically modular and the first two and most crucial steps involve image demosaicking and image denoising. Due to the modular nature of the pipeline, demosaicking and denoising are dealt in a sequential manner where the ordering will either alter the light intensity readings from the sensor if denoising will be applied first, or the initial demosaicking will introduce non-linearities in the noise statistics rending denoising an even harder problem. Moreover, both of these problems belong to the category of ill-posed problems while their joint treatment is very challenging since two-thirds of the underlying data are missing and the rest are perturbed by noise. Evidently, reconstruction errors during this early stage of the camera pipeline will produce unsatisfying final results.

Since, demosaicking is an essential step of the camera pipeline, it has been extensively studied. For a complete survey of recent approaches, we refer to [109]. One of the main drawbacks of the currently introduced methods that deal with the demosaicking problem, is that they assume a specific Bayer pattern [109, 110, 111, 112, 113, 114, 115]. This is a rather strong assumption and limits their applicability since there are many cameras available in the market that employ different Color filter Array (CFA) patterns, for example Fuji sensors. Therefore, demosaicking methods that are agile and able to generalize to different CFA patterns are preferred.

Typical methods that work for any CFA are nearest neighbor and bilinear interpolation of the neighboring values for a given pixel for each channel. The problem with these approaches are the produced zippering artifacts which occur along high frequency signal changes, e.g., edges and textures. Other classic approaches make use of the self-similarity and redundancy properties of natural images [112, 110, 111, 114] in order to reconstruct the image, but they require an excessive amount of computation time and thus reducing their applicability to low-resource devices. Another successful class consists of methods that act upon the frequency domain, where any Bayer CFA can be represented as the combination of a luminance component at baseband and two modulated components [116].

During recent years, research is directed towards learning based approaches, although a common problem with the design of learning based demosaicking al-

gorithms is the lack of ground-truth images. In many approaches such as those in [117, 118, 115] the authors used already processed images as references that are simulated mosaicked again, i.e. they apply a mosaick mask on the already demosaicked images, therefore obtaining non-realistic pairs for tuning trainable methods. Under this training strategy, the main issue is that demosaicking artifacts on the training data will hinder the performance and the overall quality of the reconstruction. In a recent work Khasabi et al. [119] proposed a way to produce a dataset with realistic reference images allowing for the design of machine learning demosaicking algorithms. In their work, they thoroughly explained a methodology to create a demosaick dataset which is on par with the reality. We use the Microsoft Demosaicking dataset [119] in order to train, evaluate and compare our system. The reason is that the contained images have to be demosaicked in the linear RGB (lin-RGB) color space of the camera before being transformed via color transformation and gamma correction into standard RGB (sRGB) space that common consumer display devices use. Furthermore, two popular CFA patterns are contained into the dataset, namely the Bayer and Fuji X Trans, which permits the development and evaluation of methods that are able to deal with different CFA patterns.

The effectiveness of neural networks for image demosaicking has been studied for over a decade. In earlier works [120, 121] feed forward neural networks were used on par with dictionary methods in order to obtain adaptive solutions for image demosaicking, while in [122] small patches were used to train a multi-layer neural network minimizing an error function. In a recent work, Gharbi et. al. [123] exploit the advantages in the field of deep learning to create a deep Convolutional Neural Network (CNN) that is able to demosaick images and a lot of effort was put by the authors to create a new large demosaicking dataset, namely the MIT Demosaicking Dataset which consists of 2.6 million patches of images. Consequently, new CNN approaches were developed extending the usage of CNNs in the field. In Tan et al. [115] an ensemble of CNNs was developed which contained different models trained to demosaick patches with specific attributes, for example textures and smooth areas, while Henz et al. [124] constructed a convolutional autoencoder

which was able to jointly design CFA and demosaick, therefore they obtained a CFA that out-performed the common Bayer CFA for image reconstruction purposes.

Apart from the demosaicking problem, another problem that requires special attention is the elimination of noise arising from the sensor and which distorts the acquired raw data. Firstly, the sensor recording is corrupted with shot noise [125] which is the result of random variation of the detected photons. Second, electronic inefficiencies during reading and converting electrical charge into a digital count exhibit another type of noise, namely read noise. While shot noise is generally attributed to follow a Poisson distribution and the read noise a Gaussian distribution; under certain circumstances both noises can be approximated by a random variable following a heteroscedastic Gaussian pdf [125]. Prior work from Kalevo and Rantanen [126], analyzed whether denoising should occur before or after the demosaicking step. It was experimentally confirmed that denoising is preferably done before demosaicking, however, Farsiu et al. [127] formulates the solution of a joint estimation process in one step and demonstrates the superiority to an approach that breaks the problem into individual step. Later on, many researchers [128, 129, 130] validated experimentally the advantage of a joint estimation. Motivated by this long-known fact, we also pursue a joint approach for denoising and demosaicking of raw sensor data.

In this chapter, we propose an iterative neural network for solving the joint denoising and demosaicking problem that yielded state-of-the art results on various datasets, both real and synthetic, without the need of millions of training images. We perform an extensive line of experimentation and ablation studies to identify the performance of the proposed method with different configurations such as the number of iterations and the effect of different initialization schemes in the overall quality of the reconstruction.

2.2 Problem Formulation

To solve the joint demosaicking-denoising problem, one of the most frequently used approaches in the literature relies on the following linear observation model

$$\mathbf{y} = \mathbf{M}\mathbf{x} + \mathbf{n}, \tag{2.1}$$

which relates the observed sensor raw data, $\mathbf{y} \in \mathbb{R}^N$, and the underlying image $\mathbf{x} \in \mathbb{R}^N$ that we aim to restore. Both $\mathbf{x}$ and $\mathbf{y}$ correspond to the vectorized forms of the images assuming that they have been raster scanned using a lexicographical order. Under this notation, $\mathbf{M} \in \mathbb{R}^{N \times N}$ is the degradation matrix that models the spatial response of the imaging device, and in particular the CFA pattern. According to this, $\mathbf{M}$ corresponds to a square diagonal binary matrix where the zero elements in its diagonal indicate the spatial and channel locations in the image where color information is missing. Apart from the missing color values, the image measurements are also perturbed by noise which hereafter, we will assume that is an i.i.d Gaussian noise $\mathbf{n} \sim N(0, \sigma^2)$. Note, that this is a rather simplified assumption about the noise statistics distorting the measurements. Nevertheless, our derived data-driven algorithm will be trained and evaluated on images that are distorted by noise, which follows statistics that better approximate real noisy conditions [125].

Recovering $\mathbf{x}$ from the measurements $\mathbf{y}$ belongs to the broad class of linear inverse problems. For the problem under study, the operator $\mathbf{M}$ is clearly singular, i.e. not invertible. This fact combined with the presence of noise perturbing the measurements leads to an ill-posed problem where a unique solution does not exist. One popular way to deal with this, is to adopt a Bayesian approach and seek for the Maximum A Posteriori (MAP) estimator

$$\begin{aligned} \mathbf{x}^\star &= \arg\max_{\mathbf{x}} \log(p(\mathbf{x}|\mathbf{y})) \\ &= \arg\max_{\mathbf{x}} \log(p(\mathbf{y}|\mathbf{x})) + \log(p(\mathbf{x})), \end{aligned} \tag{2.2}$$

where $\log(p(\mathbf{y}|\mathbf{x}))$ represents the log-likelihood of the observation $\mathbf{y}$ and $\log(p(\mathbf{x}))$ represents the log-prior of $\mathbf{x}$. Problem (2.2) can be equivalently re-casted as the

minimization problem

$$\mathbf{x}^{\star} = \arg\min_{\mathbf{x}} \frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{M}\mathbf{x}\|_2^2 + \phi(\mathbf{x}), \tag{2.3}$$

where the first term corresponds to the negative log-likelihood (assuming i.i.d Gaussian noise of variance σ^2) and the second term corresponds to the negative log-prior. According to the above, the restoration of the underlying image $\mathbf{x}$, boils down to computing the minimizer of the objective function in Eq. (2.3), which consists of two terms. This problem formulation has also direct links to variational methods where the first term can be interpreted as the data-fidelity that quantifies the proximity of the solution to the observation and the second term can be seen as the regularizer, whose role is to promotes solutions that satisfy certain favorable image properties.

In general, the minimization of the objective function

$$Q(\mathbf{x}) = \frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{M}\mathbf{x}\|_2^2 + \phi(\mathbf{x}) \tag{2.4}$$

is far from a trivial task, because the solution cannot simply be obtained by solving a set of linear equations. From the above, it is now clear that there are two important challenges that need to be dealt with before we are in position of deriving a satisfactory solution for our problem. The first one is to come up with an algorithm that can efficiently minimize $Q(\mathbf{x})$, while the second one is to select an appropriate form for $\phi(\mathbf{x})$, which will constrain the set of admissible solutions by promoting only those that exhibit the desired properties.

In Section 2.3, we focus on the first challenge, while in Section 2.4 we discuss how it is possible to avoid making any explicit decisions for the regularizer (or equivalently the negative log-prior) by following a machine learning approach. As we will descirbe in detail, the proposed strategy will allow us to efficiently regularize our solution but without the need to explicitly learn the form of the regularizer $\phi(\mathbf{x})$.

2.3 Majorization Minimization Method

One of the main difficulties in the minimization of the objective function in Eq. (2.4) is the coupling that exists between the singular degradation operator, $\mathbf{M}$, and the latent image $\mathbf{x}$. To circumvent this difficulty there are several optimization strategies available that we could use, with potential candidates being splitting variables techniques such as the Alternating Direction Method of Multipliers [131] and the Split Bregman approach [132]. However, one difficulty that arises by using such methods is that they involve additional parameters that need to be tuned so that a satisfactory convergence speed to the solution is achieved. Unfortunately, there is not a simple and straightforward way to choose these parameters. For this reason, in this work we will instead pursue a majorization-minimization (MM) approach [133, 134, 135], which does not pose such a requirement. Under this framework, as we will describe in detail, instead of obtaining the solution by minimizing (2.4), we compute it iteratively via the successive minimization of surrogate functions. The surrogate functions provide an upper bound of the initial objective function [133] and they are simpler to deal with than the original objective function.

Specifically, in the majorization-minimization (MM) framework, an iterative algorithm for solving the minimization problem

$$\mathbf{x}^{*} = \arg\min_{f} Q\left(\mathbf{x}\right) \tag{2.5}$$

takes the form [133]

$$\mathbf{x}^{(t+1)} = \arg\min_{x} \tilde{Q}(\mathbf{x};\mathbf{x}^{(t)}), \tag{2.6}$$

where $\tilde{Q}(\mathbf{x};\mathbf{x}^{(t)})$ is the majorizer of the function $Q(\mathbf{x})$ at a fixed point $\mathbf{x}^{(t)}$, satisfying the two conditions

$$\tilde{Q}(\mathbf{x};\mathbf{x}^{(t)}) > Q(\mathbf{x}), \forall \mathbf{x} \neq \mathbf{x}^{(t)} \quad \text{and} \quad \tilde{Q}(\mathbf{x}^{(t)};\mathbf{x}^{(t)}) = Q(\mathbf{x}^{(t)}). \tag{2.7}$$

Here, the underlying idea is that instead of minimizing the actual objective function $Q(\mathbf{x})$, we fist upper-bound it by a suitable majorizer $\tilde{Q}(\mathbf{x};\mathbf{x}^{(t)})$, and then

minimize this majorizing function to produce the next iterate $\mathbf{x}^{(t+1)}$. Given the properties of the majorizer, iteratively minimizing $\tilde{Q}(\cdot;\mathbf{x}^{(t)})$ also decreases the objective function $Q(\cdot)$. In fact, it is not even required that the surrogate function in each iteration is minimized. It is sufficient to find a $\mathbf{x}^{(t+1)}$ that only decreases it.

To derive a majorizer for $Q(\mathbf{x})$ we opt for a majorizer of the data-fidelity term (negative log-likelihood). In particular, we consider the following majorizer

$$\tilde{d}(\mathbf{x},\mathbf{x}_0) = \frac{1}{2\sigma^2}\|\mathbf{y}-\mathbf{M}\mathbf{x}\|_2^2 + d(\mathbf{x},\mathbf{x}_0), \tag{2.8}$$

where $d(\mathbf{x},\mathbf{x}_0) = \frac{1}{2\sigma^2}(\mathbf{x}-\mathbf{x}_0)^T[\alpha\mathbf{I}-\mathbf{M}^T\mathbf{M}](\mathbf{x}-\mathbf{x}_0)$ is a function that measures the distance between $\mathbf{x}$ and $\mathbf{x}_0$. Since $\mathbf{M}$ is a binary diagonal matrix, it is an idempotent matrix, that is $\mathbf{M}^T\mathbf{M}=\mathbf{M}$, and thus $d(\mathbf{x},\mathbf{x}_0) = \frac{1}{2\sigma^2}(\mathbf{x}-\mathbf{x}_0)^T[\alpha\mathbf{I}-\mathbf{M}](\mathbf{x}-\mathbf{x}_0)$. According to the conditions in (2.7), in order $\tilde{d}(\mathbf{x},\mathbf{x}_0)$ to be a valid majorizer, we need to ensure that $d(\mathbf{x},\mathbf{x}_0)\geq 0, \forall\mathbf{x}$ with equality iff $\mathbf{x}=\mathbf{x}_0$. This suggests that $a\mathbf{I}-\mathbf{M}$ must be a positive definite matrix, which only holds when $\alpha > \|\mathbf{M}\|_2 = 1$, i.e α is bigger than the maximum eigenvalue of the binary diagonal matrix $\mathbf{M}$. Based on the above, the upper-bounded version of (2.4) is finally written as

$$\tilde{Q}(\mathbf{x},\mathbf{x}_0) = \frac{1}{2(\sigma/\sqrt{a})^2}\|\mathbf{x}-\mathbf{z}\|_2^2 + \phi(\mathbf{x}) + c, \tag{2.9}$$

where c is a constant and $\mathbf{z}=\mathbf{y}+(\mathbf{I}-\mathbf{M})\mathbf{x}_0$.

Notice that following this approach, we have managed to completely decouple the degradation operator $\mathbf{M}$ from $\mathbf{x}$ and we now need to deal with a simpler problem. In fact, the resulting surrogate function in Eq. (2.9) can be interpreted as the objective function of a denoising problem, with $\mathbf{z}$ being the noisy measurements that are corrupted by noise whose variance is equal to σ^2/a. This is a key observation that we will heavily rely on, in order to design our deep network architecture. In particular, now it is possible instead of selecting the form of $\phi(\mathbf{x})$ and minimizing the surrogate function, to employ a denoising neural network in order to compute the solution of the current iteration.

Our idea is similar in nature to other recent image restoration approaches

that have employed denoising networks as part of alternative iterative optimization strategies, such as RED [136], P^3 [137] and IRCNN [138]. However, a important difference is that in our case we completely avoid the introduction of any free parameter that needs to be tuned by the user. We also move one step further from the aforementioned approaches by finetuning our denoising neural network on available data which allows us to implicitly incorporate domain knowledge to our learned regularizer. This direction for solving the joint denoising-demosaicking problem is very appealing since by using training data we can implicitly learn the function $\phi(\mathbf{x})$ and also minimize the corresponding surrogate function using a feed-forward network.

2.4 Residual Denoising Network (ResDNet)

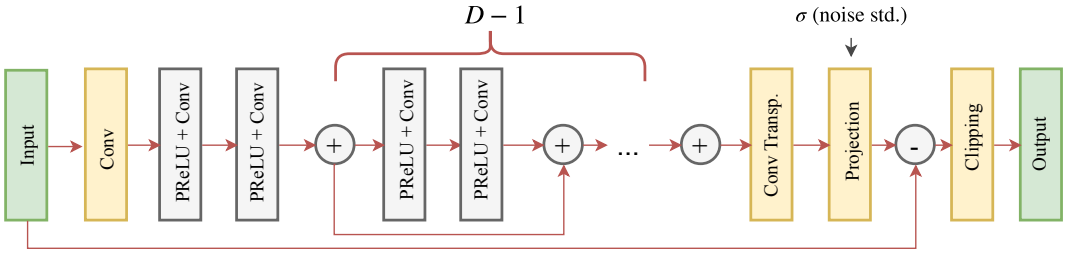


Figure 2.1: Illustration of the residual denoiser; a core module of our framework. The deep learning based denoiser is used in every iteration to provide an approximate solution of the surrogate function.

Based on the discussion above, an important part of our approach is the design of a denoising network that will play the role of the solver for the surrogate function in Eq. (2.9). The architecture of the proposed network is depicted in Fig. 2.1. This is a residual network similar to DnCNN [139], where the output of the network is subtracted from its input. Therefore, the network itself acts as a noise estimator and its task is to estimate the noise realization that distorts the input. Such network architectures have been shown to lead to better restoration results than alternative approaches [139, 140].

One distinctive difference between our network and DnCNN, which also makes our network suitable to be used as a part of the MM-approach, is that it

accepts two inputs, namely the distorted input and the variance of the noise. This way, as we will demonstrate in the sequel, we are able to learn a single set of parameters for our network and to employ the same network to inputs that are distorted by a wide range of noise levels. While the blind version of DnCNN can also work for different noise levels, as opposed to our network it features an internal mechanism to estimate the noise variance. However, when the noise statistics deviate significantly from the training conditions such a mechanism can fail and thus DnCNN can lead to poor denoising results. In fact, due to this reason in [138], where more general restoration problems than denoising have been studied, the authors of DnCNN use a non-blind variant of their network as a part of their proposed restoration approach. However, training a deep network that requires a large number of parameters to be learned for each noise level can be rather impractical, especially in cases where one would like to employ such networks on devices with limited storage capacities.

In our case, inspired by the recent work in [140] we circumvent this limitation by explicitly providing as input to our network the noise variance, which is then used to assist the network so as to provide an accurate estimate of the noise distorting the input. Note that there are several techniques available in the literature that can provide an estimate of the noise variance, such as those described in [141, 142], and thus this requirement does not pose any significant challenges in our approach.

A ResDNet with depth D, consists of five fundamental blocks. The first block is a convolutional layer with 64 filters whose kernel size is 5×5. The second one is a non-linear block that consists of a parametrized rectified linear unit activation function (PReLU), followed by a convolution with 64 filters of 3×3 kernels. The PReLU function is defined as $\text{PReLU}(\mathbf{x}) = \max(0,\mathbf{x}) + \boldsymbol{\kappa} * \min(0,\mathbf{x})$ where $\boldsymbol{\kappa}$ is a vector whose size is equal to the number of input channels. In our network we use $D*2$ distinct non-linear blocks which we connect via a shortcut connection every second block in a similar manner to [143] as shown in Fig. 2.1. Next, the output of the non-linear stage is processed by a transposed convolution layer which reduces the number of channels from 64 to 3 and has a kernel size of 5×5. Then, it follows a projection layer [140] which accepts as an additional input the noise variance and

whose role is to normalize the noise realization estimate so that it will have the correct variance, before this is subtracted from the input of the network. Finally the result is clipped so that the intensities of the output lie in the range $[0,255]$. This last layer enforces our prior knowledge about the expected range of valid pixel intensities.

Regarding implementation details, before each convolution layer the input is padded to make sure that each feature map has the same spatial size as the input image. However, unlike the common approach followed in most of the deep learning systems for computer vision applications, we use reflexive padding than zero padding. Another important difference to other networks used for image restoration tasks [139, 138] is that we don't use batch normalization after convolutions. Instead, we use the parametric convolution representation that has been proposed in [140] and which is motivated by image regularization related arguments.

In particular, if $\mathbf{v} \in \mathbb{R}^L$ represents the weights of a filter in a convolutional layer, these are parametrized as

$$\mathbf{v} = s\frac{(\mathbf{u}-\bar{\mathbf{u}})}{\|\mathbf{u}-\bar{\mathbf{u}}\|_2}, \tag{2.10}$$

where s is a scalar trainable parameter, $\mathbf{u} \in \mathbb{R}^L$ and $\bar{\mathbf{u}}$ denotes the mean value of $\mathbf{u}$. In other words, we are learning zero-mean valued filters whose ℓ_2-norm is equal to s.

Furthermore, the projection layer, which is used just before the subtraction operation with the network input, corresponds to the following ℓ_2 orthogonal projection

$$\mathbb{P}_{\mathbb{C}}(\mathbf{y}) = \varepsilon\frac{\mathbf{y}}{\max(\|\mathbf{y}\|_2, \varepsilon)}, \tag{2.11}$$

where $\varepsilon = e^{\gamma}\theta$, $\theta = \sigma\sqrt{N-1}$, N is the total number of pixels in the image (including the color channels), σ is the standard deviation of the noise distorting the input, and γ is a scalar trainable parameter. As we mentioned earlier, the goal of this layer is to normalize the noise realization estimate so that it has the desired variance before it is subtracted from the network input.

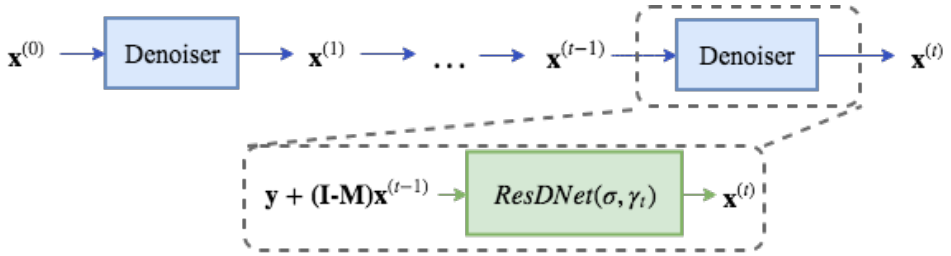


Figure 2.2: A graphical representation of the proposed iterative neural network. We have omitted the extrapolation steps for clarity.

2.5 Demosaicking Network Architecture

The overall architecture of our approach is based upon the MM framework, presented in Section 2.3, and the proposed denoising network. As discussed, the MM is an iterative algorithm Eq. (2.6) where the minimization of the majorizer in Eq. (2.9) can be interpreted as a denoising problem. One way to design the demosaicking network would be to unroll the MM algorithm as K discrete steps and then for each step use a different denoising network to retrieve the solution of Eq. (2.9). However, this approach can have two distinct drawbacks which will hinder its performance. The first one, is that the usage of a different denoising neural network for each step like in [138], demands a high overall number of parameters, which is equal to K times the parameters of the employed denoiser, making the demosaicking network impractical for any real applications. Simultaneously, the high overall number of network parameters would require a significantly higher number of training data and training time. To override these drawbacks, we opt to use our ResDNet denoiser, which can be applied to a wide range of noise levels, for all K steps of our demosaick network, using the same set of parameters. By sharing the parameters of our denoiser across all the K steps, the overall demosaicking approach maintains a low number of parameters and requires only a few hundred of images to train.

The second drawback of the MM framework as described in Section 2.3 is the slow convergence [144] that it can exhibit. Beck and Teboulle [144] introduced an accelerated version of this MM approach which combines the solutions of two consecutive steps with a certain extrapolation weight that is different for every step. In this work, we adopt a similar strategy which we describe in Algorithm 1. Fur-

Algorithm 1: The proposed demosaicking network described as an iterative process. The ResDNet parameters are shared across all iterations.

Input: $\mathbf{M}$: CFA, $\mathbf{y}$: input, K: iterations, $\boldsymbol{w} \in \mathbb{R}^K$: extrapolation weights, σ: estimated noise, $\boldsymbol{\gamma} \in \mathbb{R}^K$: projection parameters

$\mathbf{x}^{(0)} = \mathbf{0}$;

Initialize $\mathbf{x}^{(1)}$ using $\mathbf{y}$;

for $i \leftarrow 1$ **to** K **do**

  $\mathbf{u} = \mathbf{x}^{(i)} + \boldsymbol{w}_i(\mathbf{x}^{(i)} - \mathbf{x}^{(i-1)})$;

  $\mathbf{x}^{(i+1)} = \text{ResDNet}((\mathbf{I} - \mathbf{M})\mathbf{u} + \mathbf{y}, \sigma, \boldsymbol{\gamma}_i)$;

end

thermore, in our approach we go one step further and instead of using the values originally suggested in [144] for the weights $\boldsymbol{w} \in \mathbb{R}^K$, we treat them as trainable parameters and learn them directly from the data. These weights are initialized with $w_i = \frac{i-1}{i+2}, \forall 1 \leq i \leq K$. The underlying reasoning of finetuning the extrapolation weights lies in [145] where the authors claim that some extrapolation weights may actually hinder the convergence, so we opt to fit the weights upon available data and derive a data-driven extrapolation suitable for our problem.

The convergence of our framework can be further sped up by employing a continuation strategy [146] where the main idea is to solve the problem in Eq. (2.9) with a large value of σ and then gradually decrease it until the target value is reached. Our approach is able to make use of the continuation strategy due to the design of our ResDNet denoiser, which accepts as additional arguments the noise variance that is fed to the learnable projection layer, and the ability to denoise images for various noise levels. In detail, we initialize the trainable parameter of the projection layer $\boldsymbol{\gamma} \in \mathbb{R}^K$ with values spaced evenly on a log scale from γ_{max} to γ_{min} and later on the vector $\boldsymbol{\gamma}$ is further finetuned on the training dataset via back-propagation.

In summary, our overall demosaicking network is described in Algorithm 1 where the set of trainable parameters θ consists of the parameters of the ResDNet denoiser, the extrapolation weights $\boldsymbol{w}$ and the projection parameters $\boldsymbol{\gamma}$. All of the aforementioned parameters are initialized as described in the current section and Section 2.4 and are trained on specific demosaick datasets.

Finally, while our demosaick network shares a similar philosophy with meth-

ods such as RED [136], P^3 [137] and IRCNN [138], it exhibits some important and distinct differences. In particular, the aforementioned strategies make use of certain optimization schemes to decompose their original problem into sub-problems that are solvable by a denoiser. For example, the authors of P^3 [137] decompose the original problem Eq. (2.1) relying on the Alternating Direction Method of Multipliers (ADMM) [131] and solve instead a linear system of equations and a denoising problem. The authors of RED [136] go one step further and employ a regularizer that involves an image-adaptive Laplacian, which in turn allows the use of several classical and machine-learning based denoisers to serve as sub-solvers. Both approaches are similar to ours, however their formulation involves a tunable variable λ that weights the participation of the regularizer on the overall optimization procedure. Thus, in order to obtain an accurate reconstruction in reasonable time, the user must manually tune the variable λ which is tractable but not a trivial task. On the other hand, our method does not involve any tunable variables. Furthermore, the approaches P^3, RED and IRCNN are based upon static denoisers like Non Local Means [147], BM3D [148] and DCNN [139], meanwhile we opt to use a universal denoiser, like ResDNet, that can be further trained on any available training data. Finally, our approach goes one step further and we use a trainable version of an iterative optimization strategy for the task of the joint denoising-demosaicking in the form of a feed-forward neural network.

2.6 Network Training

2.6.1 Joint Denoising and Demosaicking

Since Eq. (2.9) is, as already discussed, a denoising step, we pre-train our ResDNet denoiser on the simple case where $\mathbf{M} = \mathbf{I}$; casting our problem an Additive White Gaussian(AWGN) denoising task. We found experimentally that pre-training the ResDNet vastly reduces the necessary training time because it is a good initialization for the joint denoising and demosaicking task. In detail, the denoising network ResDNet that we use as part of our overall network is pre-trained on the Berkeley segmentation dataset (BSDS) [149], which consists of 500 color images. These

images were split in two sets, 400 were used to form a train set and the rest 100 formed a validation set. All the images were randomly cropped into patches of size 180×180 pixels. The patches were perturbed with noise $\sigma \in [0, 15]$ and the network was optimized to minimize the Mean Square Error. We set the network depth $D = 5$, all weights are initialized as in He et al. [150] and the optimization is carried out using AMSGRAD [151] which is a stochastic gradient descent algorithm that adapts the learning rate per parameter. The training procedure starts with an initial learning rate equal to 10^{-2}.

Using the pre-trained denoiser, our overall network is further trained end-to-end to minimize the averaged L_1 loss over a mini-batch of size d.

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{d} \|\mathbf{y}_i - f(\mathbf{x}_i)\|_1 , \tag{2.12}$$

where $\mathbf{y}_i \in \mathbb{R}^N$ and $\mathbf{x}_i \in \mathbb{R}^N$ are the rasterized ground-truth and input images, while $f(\mathbf{x}_i)$ denotes the output of our proposed network. Due to the iterative nature of our framework, the network parameters are updated using the Back-propagation Through Time (BPTT) algorithm. Specifically, using the same denoiser for every iteration in Alg. 1 means that the same set of parameters is used for every iteration and thus we have to sum the parameters changes in the K unfolded instances in order to train the network. However, if the number of total iterations K is high, then a number of prohibitive restrictions arise during training, for example both K and mini-batch size d are upper-bounded from the GPU memory consumption. It is evident that we need to keep all intermediate results of the Alg. 1 in order to calculate the gradients which increases the memory requirements by a factor of K. Thankfully, there is a workaround to avoid such restrictions by using instead the Truncated Back-propagation Through Time (TBPTT) [152] algorithm, which we explain in detail in Section 2.6.2. Using this solution, we are able to use an arbitrary number of total iterations K for training and increase the batch size d with the only trade-off of being the increase in the computation time. Furthermore, the optimization is carried again via the AMSGRAD optimizer and the training starts from a learning rate of 10^{-2} which we decrease it by a factor of 10 every 100 epochs.

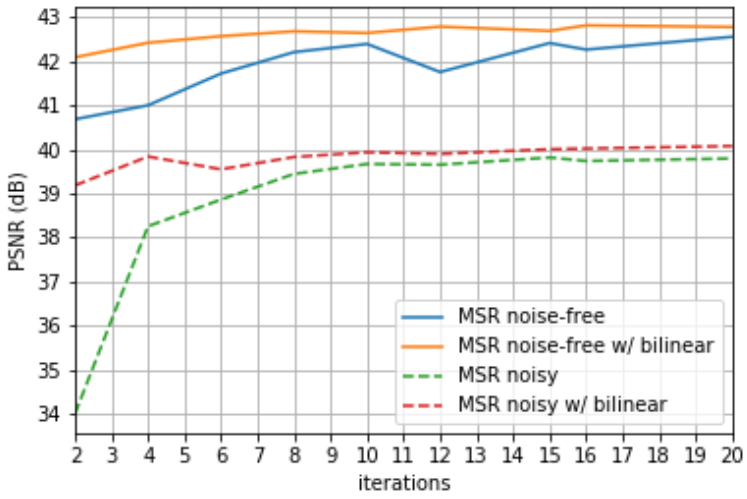


Figure 2.3: Increasing the number of iterations, our deep network is capable to achieve increasingly better reconstruction with the same number of parameters. Also, it can be seen that given a large number of iterations, the method is capable to achieve virtually the same performance with and without proper initialization.

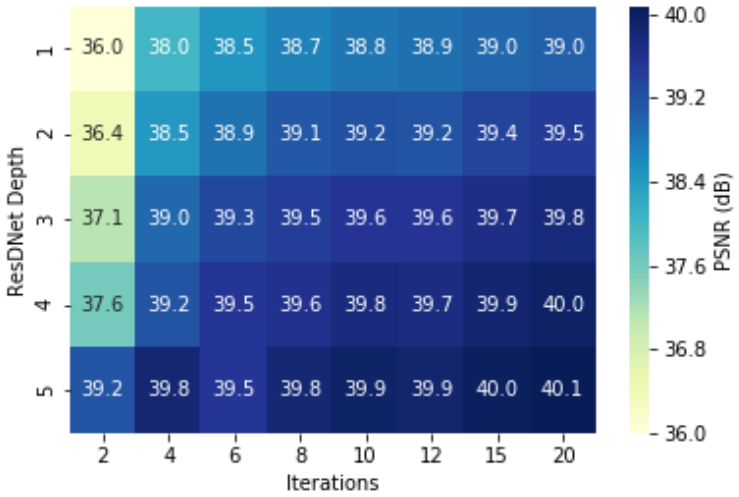


Figure 2.4: Results comparing the depth of the denoiser versus the number of iteration on the noisy MSR Dataset. The heat-map depicts the ability of our method to generalize by decreasing the number of parameters and simultaneously increasing the number of iterations.

Finally, for every noise-free experiment we set $\gamma_{max} = 15$ and $\gamma_{min} = 0$, while for every other case the respective values are $\gamma_{max} = 2$ and $\gamma_{min} = 0$.

2.6.2 Training with TBPTT

As discussed, we would like to use an arbitrary amount of iterations K during training, however this is not always possible because K is upper-bounded by the available GPU memory or RAM. Consequently, a higher number of iterations force us to use smaller batches casting the training of our network slow. Therefore, to overcome this restriction we propose to train our network with the TBPTT algorithm where the loop is unrolled into a small set of k iterations (stages) out of K and the back-propagation is performed sequentially on the unrolled stages. Thankfully, TBPPT comes with no additional computational cost because an optimized implementation of TBPTT has the same computation complexity with the full BPTT.

In similar learnable iterative approaches such as those in [153, 85] the authors argued that a few number of iterations between 4 to 6 are enough for obtaining a good reconstruction quality. However, we experimentally found that while a few number of iterations might lead to adequate results aimed mostly on applications where computational time needs to be kept relatively low, the best results can be achieved with as many as 20 iterations or even more. Indeed, as presented in Figs. 2.3 and 2.4 the Peak Signal-to-Noise Ratio (PSNR) is increasing till a certain number of iterations and after that it stabilizes. Also, from Fig. 2.3 it is clear that a proper initialization of the network's input allows the use of only a few iterations in order to retrieve comparable performance.

Nevertheless, our method is capable to achieve the same performance even with an improper initialization if a higher number of iterations is used, casting our approach very attractive for other inverse problems where no good initialization exist such as compressed sensing. Finally, in Fig. 2.4 the trade-off between the number of network parameters and the iterations is demonstrated. In particular, from these results we observe that networks which involve a denoising network of smaller depth and thus less trainable parameters, can produce meaningful results if the number of iterations is high enough, while networks that depend on deeper denoising networks require only a small number of iterations in order to match the same performance.

2.7 Experiments

We perform an extensive line of experimentation on multiple datasets and CFA patterns in order to evaluate and analyze our method. Our main metric used for comparisons among different methods in all reported experiments is the PSNR.

2.7.1 Demosaicking Artificial Data

At first, we compare our method against prior work on the pure demosaicking task with artificially created mosaicked data on the *sRGB* color space. The artificial data are created using the standard datasets McMaster [110], Kodak [109] and the newly created MIT dataset [123]; all of which are 8-bit sRGB data and they are re-mosaicked, so the following experiments deviate from the standard camera pipeline used in practical scenarios. Beyond this, the aforementioned dataset are known to have flaws [123] and misrepresent the statistics of natural images [154]. Due to the fact that these images are noise-free, we manually set $\sigma = 1$ as the noise standard deviation for all images and no noise estimation takes place. The whole MIT Dataset is used for training for 10 epochs and we set $K = 1$ in order to speed up the training process; after that, the network is trained on a small random subset of 40.000 with $K = 10$. In this case, the input is the mosaicked image and no initialization takes place. From the reported results in Table 2.1 we observe that our network achieves comparable performance for all different datasets with current state-of-the art approaches, although, it requires only a fraction of the parameters that the other systems use.

2.7.2 Demosaicking Raw Data

As mentioned earlier, Khashabi et al. [119] proposed that the evaluation of demosaicking and denoising should occur on raw RGB data, because this testing pipeline is closely related to real digital imaging applications. MSR dataset contains exclusively linear data encoded in the standard imaging 16-bit representation. The dataset contains 200 images for training, which we augment with vertical and horizontal flips, 100 for validation purposes and a test set of 200 images. The same 200 training images are provided with noise perturbations using the affine noise

Table 2.1: Quantitative Comparison of our system to state-of-the-art techniques on the demosaick-only scenario on artificial data in terms of Peak signal-to-noise ratio (PSNR) performance.

	Kodak	McM	VDP	Moire
Bilinear	32.9	32.5	25.2	27.6
Zhang (NLM) [110]	37.9	36.3	30.1	31.9
Hirakawa [155]	36.5	33.9	30.0	32.1
Getreuer [156]	38.1	36.1	30.8	32.5
Heide [113]	40.0	38.6	27.1	34.9
Klatzer [130]	35.3	30.8	28.0	30.3
Gharbi [123]	41.2	39.5	34.3	37.0
Kokkinos [157]	41.5	**39.7**	**34.5**	37.0
Tan [115]	**42.0**	39.0	-	-
Henz [124]	41.9	39.5	34.3	36.3
MMNet_{10} (ours)	**42.0**	**39.7**	**34.5**	**37.1**

model [125] with unknown values for the hyper-parameters of the noise. The lack of these hyperparameters renders difficult the production of training data that follow the same noise statistics. This reason, forced Gharbi et al. [123] to use a simpler approach for Gaussian noise which diverges greatly from practical applications and this fact is reflected from the inferior performance that their system achieves in the noisy case (38.6 dB). While me made the same simplistic assumption in Section 2.3 in order to design the network architecture, our network after being trained on more realistic noise conditions is capable to denoise successfully even the data dependent part of the affine noise models.

We estimate σ for each raw image using the median absolute deviation of the wavelet detail coefficients as described in [158] and provide it to the network as additional input. Surprisingly, as show in Table 2.2 our method is capable to achieve slightly better performance to Gharbi et al. in the noise-free scenario using only 200 training images and surpass previous approaches in the noisy scenario by a large margin of 1.3 dB if we employ $K = 20$ iterations and 0.4 dB if we set $K = 2$. This fact clearly demonstrates the capabilities of our method to generalize better even when trained on small datasets and the ability to trade off performance for computing time and vice versa.

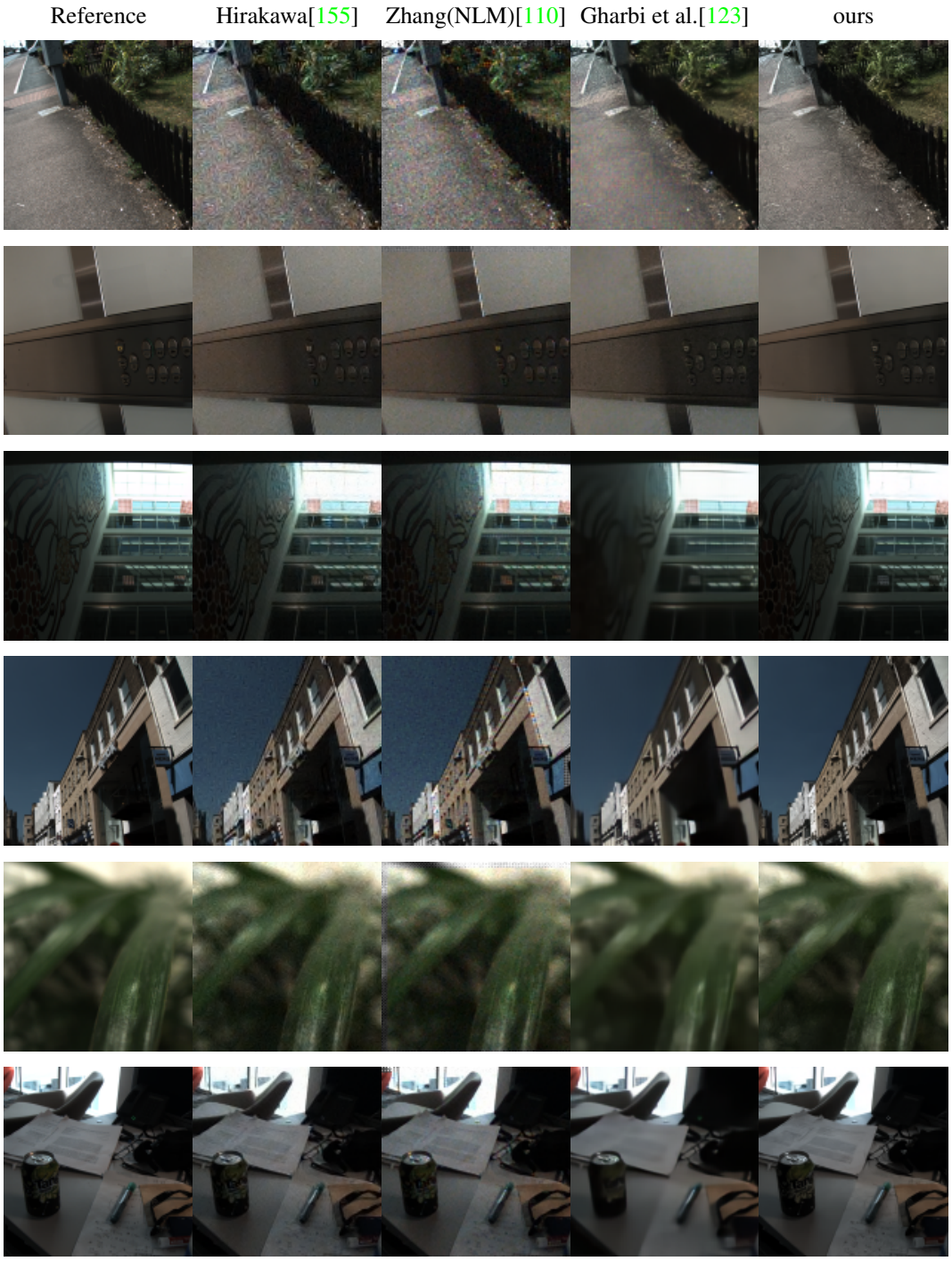


Figure 2.5: Comparison of our network with other competing techniques on images from the noisy MSR Dataset. From these results it is clear that our method is capable of removing the noise while keeping fine details. On the contrary, the rest of the methods either fail to denoise or they oversmooth the images.

Table 2.2: PSNR performance of different methods in both linear and sRGB spaces. The results of methods that cannot perform denoising are not included for the noisy scenario. The color space in the brackets indicates the particular color space of the employed training dataset. We exclude PSNR scores of methods who are pure demosaicking from the noisy columns.

	noisy		noise-free	
	linRGB	sRGB	linRGB	sRGB
Bilinear	-	-	30.9	24.9
Zhang(NLM) [110]	-	-	38.4	32.1
Hirakawa [155]	-	-	37.2	31.3
Getreuer [156]	-	-	39.4	32.9
Heide [113]	-	-	40.0	33.8
Khasabi [119]	37.8	31.5	39.4	32.6
Klatzer [130]	38.8	32.6	40.9	34.6
Gharbi [123]	38.6	32.6	42.7	35.9
Kokkinos [157]	39.2	33.3	41.0	34.6
MMNet_2 (ours)	39.2	33.3	42.1	35.6
MMNet_{20} (ours)	**40.1**	**34.2**	**42.8**	**36.4**

Table 2.3: Evaluation on noise-free linear data with the non-Bayer mosaick pattern Fuji XTrans.

	noise-free	
	linear	sRGB
Khashabi [119]	36.9	30.6
Klatzer [130]	39.6	33.1
Gharbi [123]	39.7	33.2
Kokkinos [157]	39.9	33.7
MMNet_8 (ours)	40.2	34.0
MMNet_{20} (ours)	**40.6**	**34.4**

2.7.3 Demosaicking Non-Bayer CFA

Finally, we explore the applicability of our approach to other Non-Bayer CFA, namely the Fuji-XTrans used by all modern Fuji digital cameras. Obviously, methods capable to demosaick images from any camera sensor are preferred for practical applications. The images contained on MSR are also provided with the Fuji Xtrans CFA but only on the noise-free case. Using this available data, we trained our method without initialization and our results are provided in Table. 2.3. Clearly, our method is capable to outperform the state of the art by a 0.7 dB margin for $K = 20$

while being trained with only 200 images with flipping augmentations. At the same time, our network is capable to outperform previous approaches even when employing only $K = 8$ iterations and using as input the mosaicked image without any proper initialization.

2.8 Conclusions

In this chapter, we presented a deep learning system that produces high-quality images from raw sensor. The proposed iterative network yields superior results both quantitative and qualitative compared to the current state-of-the-art network. Meanwhile, our approach is able to generalize well even when trained on small datasets and the number of our network parameters is kept low compared to other competing solutions. Finally, we introduce an efficient way to train iterative networks with arbitrary number of iterations, which we hope that will pave the way to successful training of learning-based iterative approaches for other similar image restoration tasks.

Chapter 3

Self-supervised 3D reconstruction of articulated categories

In this chapter we propose a method that uses self-supervision to 3D reconstruct highly non-rigid objects from images and videos. Monocular 3D reconstruction of articulated object categories is challenging due to the lack of training data and the inherent ill-posedness of the problem. We use video self-supervision, forcing the consistency of consecutive 3D reconstructions by a motion-based cycle loss. This largely improves both optimization-based and learning-based 3D mesh reconstruction. We further introduce an interpretable model of 3D template deformations that controls a 3D surface through the displacement of a small number of local, learnable handles. We formulate this operation as a structured layer relying on mesh-laplacian regularization and show that it can be trained in an end-to-end manner. We finally introduce a per-sample numerical optimisation approach that jointly optimises over mesh displacements and cameras within a video, boosting accuracy both for training and also as test time post-processing. While relying exclusively on a small set of videos collected per category for supervision, we obtain state-of-the-art reconstructions with diverse shapes, viewpoints and textures for multiple articulated object categories.

The contents of this chapter were presented to the Conference on Computer Vision and Pattern Recognition (CVPR).

3.1 Introduction

Monocular 3D reconstruction of general articulated categories is a task that humans perform routinely, but remains challenging for current computer vision systems. The breakthroughs achieved for humans [28, 35, 159, 160, 161, 33, 162, 163, 164] have relied on expressive articulated shape priors [27] and mocap recordings to provide strong supervision in the form of 3D joint locations. Still, for general articulated categories, such as horses or cows, the problem remains in its infancy due to both the lack of strong supervision [37] and the inherent challenge of representing and learning articulated deformations for general categories.

Recent works have started tackling this problem by relying on minimal, 2D-based supervision such as manual keypoint annotations or masks [56] and learning morphable model priors [56, 54, 3, 59] or hand-crafted mesh segmentations [4]. In this work we leverage the rich information available in videos, and use networks trained for the 2D tasks of object detection, semantic segmentation, and optical flow to complement 2D keypoint-level supervision.

We make three contributions towards pushing the envelope of monocular 3D object category reconstruction, by injecting ideas from structure-from-motion, geometry processing and bundle adjustment in the task of monocular 3D articulated reconstruction.

Firstly, we draw inspiration from 3D vision which has traditionally relied on motion information for SFM [165, 166], SLAM [167, 168] or Non-Rigid SFM [169, 170, 171]. These category-agnostic techniques interpret 2D point trajectories in terms of an underlying 3D scene and a moving camera. In this work we use the same principle to supervise monocular 3D category reconstruction, effectively allowing us to leverage video as a source of self-supervision. In particular we establish dense correspondences between consecutive video frames through optical flow and force the back projections of the respective 3D reconstructions to be consistent with the optical flow results. This loss can be back-propagated through the 3D lifting pipeline, allowing us to supervise both the camera pose estimation and mesh reconstruction modules through video. Beyond coming for free, this su-

pervision also ensures that the resulting models will exhibit a smaller amount of jitter and be more flexible when processing videos, since the motion-based loss is sensitive to inconsistencies across consecutive frames, and failure to co-vary with moving object parts.

Secondly, we introduce a model for mesh deformations that allows for learnable, part-level mesh control, inherently accommodates mesh regularisation, and is back-propagateable, providing us with a drop-in replacement to the common morphable model paradigm adopted in [3]. For this we rely on the Laplacian surface deformation algorithm [172], commonly used in geometry processing to deform a template mesh through a set of control points ('handles') while preserving the surface structure and details. We observe that the result of this optimization-based algorithm is differentiable in its inputs, i.e. can be used as a structured layer. We incorporate this operation as the top layer of a deep network tasked with regressing the position of the control points given an RGB image. Our results show that we can learn meaningful control points that allow us to capture limb articulations while also providing a human-interpretable interface that enables the manual post-processing and further refinement using any available 3D software.

Thirdly we adopt an optimization-based approach to 3D reconstruction that is inspired from bundle adjustment [173]: given a video, we use the 'bottom-up' reconstructions of consecutive frames delivered by our CNN in terms of cameras and handle positions as the initialisation for a numerical optimisation algorithm. We then jointly optimise the per-frame mask and/or keypoint reprojection losses, and video-level motion consistency losses with respect to the cameras and handle variables, giving a 'top-down' refinement of our solution that better matches the image evidence. We show that this serves as a method for improving the results at test-time based on whatever image evidence can be obtained without manual annotation.

We evaluate our approach on 3D shape, pose and texture reconstruction on a range of different objects that exhibit diverse articulations in nature. Our qualitative results show that our method successfully captures intricate shape deforma-

tions across instances. Our ablation highlights the importance of the employed self-supervised losses and the tolerance of our method to the number of learnable handles, while our qualitative results indicate that our method largely outperforms the results of competing approaches.

3.2 Connection to Related Work

Pose, Texture and Articulation Prediction Our work addresses the task of inferring the camera pose, articulation and texture corresponding to an input image. Prior works have addressed several aspects of this problem [3, 4, 2] with varying forms of supervision. Earlier approaches like CMR [3] treat the problem of 3D reconstruction from single images using known masks and manually labelled keypoints from single viewpoint image collections. Closer to our work is the method of Kulkarni et al. [4, 2] named Canonical Surface Mapping (CSM) which produces a 3D representation in the form of a rigid or articulated template using a 2D-to-3D cycle-consistency loss. The articulated variant of CSM [2] achieves non-rigid deformation by explicitly segmenting 3D parts of the template shape manually set prior to training the method.

Surface Deformation Deformation of 3D shapes is a ubiquitous task and it is the core component of a successful image 3D reconstruction. Recent works on monocular 3D reconstruction [3, 59] treat deformation as offsets added to mesh vertices. These offsets are conditioned on images that are fed as input to deep neural networks. Plainly relocating vertices gives rise to potential surface distortions or corrupt features and this mechanism can not be interpreted or post-processed by a human modeller.

Detail-preserving deformations have been studied in the geometry processing community [172, 68, 174, 175]. Among the developed methods there is a specific subset that rely on a sparse set of control points to achieve mesh deformation. Changing the location of the control points allows the recovery of a deformed mesh as the solution of an optimization problem [172, 68]. By revisiting the aforementioned technique we derive a method on top of the Laplacian Deforma-

tion solver [172] that is capable of learning the control points and regressing their position in the 3D space.

Video-based supervision Video has been commonly used as a source of weak supervision in the context of dense labelling tasks such as semantic segmentation [176] or dense-pose estimation [177]. Drawing on the classical use of motion for 3D reconstruction, e.g. [165, 166, 167, 168, 169, 170, 171] many recent works [178, 179, 180] have also incorporated optical flow information to supervise 3D reconstruction networks. Both in the category-specific [178, 179] and agnostic [181, 182, 180] setting, optical flow provides detailed point correspondences inside the object silhouette which can aid the prediction of object articulations and the reconstruction of the underlying 3D geometry. More recent works have leveraged videos for monocular 3D human reconstruction [162] or sparsely-supervised hand-object interactions [183] based on photometric losses. In this work we show the video is a particularly effective source of supervision for our case, where we jointly learn the category-specific shape prior and the 3D reconstructions. We also rely on robust, occlusion-sensitive optical flow networks [184] which provide a stronger source of supervision than photometric consistency, since they are both trained to be solving the aperture effect in the interior of objects and also recover large displacement vectors when appropriate.

Our approach is reminiscent of the principle of cycle consistency [185, 186, 187], where the composition of two maps is meant to result in the identity mapping (in our case the lifting-based correspondence between two images and the backward-flow between). We can understand our method as being the dual of [186], where 3D synthetic data were used to learn dense correspondences between categories; here we rely on a pre-trained optical flow network to provide correspondences that in turn help learn 3D object categories.

3.3 Method Description

Given an image our target is to perform ‘inverse graphics’, namely infer the 3D shape, camera pose, and texture of the depicted object. We have at our disposal a

single representative mesh for the category ('template'), a set of 2D annotations, such as keypoints or masks (potentially extracted by neural networks, rather than manually constructed).

In our approach during training we use videos and train per-frame inverse graphics networks while exploiting temporal information for supervision. At test time we can deploy the learned networks on a per-frame level, but can also exploit temporal information, when available, to improve the accuracy of our results through a joint optimization that is inspired by bundle adjustment.

3.3.1 Learnable Laplacian Solver

Our aim in this work is to synthesise the shape of an articulated object category by a neural network. While in broad terms we adopt the deformable template paradigm adopted by most recent works [3, 59, 63], we deviate from the morphable model-based [6] modeling of shape adopted in [3, 59, 63]. In those works shape is expressed in terms of offsets $\mathbf{\Delta}_V$ of a template shape $\mathbf{V}^* = \mathbf{\Delta}_V + \mathbf{T}$, where $\mathbf{\Delta}_V$ is delivered by the last, linear, layer of a shape decoder branch, effectively modeling shape variability as an expansion on a linear basis. Such models are well-suited to categories such as faces or cars, but for objects with part-based articulation such as quadrupeds we argue that a part-level model of deformation is more appropriate - which is also the approach routinely taken in rigged modeling in graphics. Furthermore, the linear synthesis model is non-interpretable or controllable by humans and requires careful regularization during training to recover plausible meshes.

We propose instead a deformation model where a set of learnable control points (or 'handles') deform a given template so as to minimize its non-isometric deformation (i.e. stretching or squeezing) and the network's task is to regress the positions of the handles. This model is controllable, interpretable, and regularized by design, while as our experiments show it yields systematically more accurate mesh reconstruction results.

Our algorithm builds on Laplacian surface editing techniques [172] which allow us to control a template mesh through handles while minimally distorting the template's shape. We represent the 3D shape of a category as a triangular mesh

$M = (V, F)$ with vertices $\mathbf{V} \in \mathbb{R}^{N\times 3}$ and fixed edges $F \in \mathbb{Z}^{N_f\times 3}$. Our deformation approach relies on the cotangent-based discretization $\mathbf{L} \in \mathbb{R}^{N\times N}$ of the continuous Laplace-Beltrami operator used to calculate the curvature at each vertex of a mesh [188].

We obtain our K handles $H_{1,\dots,K}$ through a learnable dependency matrix $\mathbf{A} \in \mathbb{R}_{+}^{K\times N}$ that is right-stochastic, i.e. $\sum_v \mathbf{A}_{k,v} = 1$, effectively forcing every handle to lie in the convex hull of the mesh vertices by $\mathbf{H} = \mathbf{AV}$. The network's task is phrased as regressing the handle positions, denoted as $\mathbf{\Delta}_H$. Based on those handles, we obtain the deformed mesh $\mathbf{V}^*$ as the minimum of the following quadratic loss:

$$\mathbf{V}^* = \arg\min_{\mathbf{V}} \frac{1}{2} \|\mathbf{LV} - \mathbf{LT}\|^2 + \frac{1}{2} \left\|\mathbf{AV} - \tilde{\mathbf{H}}\right\|^2, \tag{3.1}$$

where as in [172] the first term enforces the solution to respect the curvature of the template mesh, $\mathbf{LT}$, while the second one penalizes the difference between the location of the handles according to $\mathbf{V}$ and the target location, $\tilde{\mathbf{H}} = \mathbf{HT} + \mathbf{\Delta}_H$. The stationary point of (3.1) can be found by solving the following linear system:

$$(\mathbf{L}^\top\mathbf{L} + \mathbf{A}^\top\mathbf{A})\mathbf{V} = \mathbf{L}^\top\mathbf{LT} + \mathbf{A}^\top\tilde{\mathbf{H}} \tag{3.2}$$

The solution $\mathbf{V}^*$ of Eq. (3.2) can be very efficiently computed with conjugate gradients or sparse solvers. In the forward case we obtain the solution using a sparse least square solver by concatenating the two matrices $\mathbf{L}$ and $\mathbf{A}$. As it will be presented the backward operation requires a different treatment and as such in the backward operation we make use of a linear solver for PSD matrices.

We want to compute the gradients with respect to the learnable dependency matrix $\mathbf{A}$ and the handle offset $\tilde{\mathbf{H}}$. We rewrite equation (3.2) as $\mathbf{WV} = \mathbf{b}$ where $\mathbf{W} = \mathbf{L}^\top\mathbf{L} + \mathbf{A}^\top\mathbf{A}$ and $\mathbf{b} = \mathbf{L}^\top\mathbf{LT} + \mathbf{A}^\top\mathbf{H}$. The direct solution is $\mathbf{V} = \mathbf{W}^{-1}\mathbf{b}$ and $\mathbf{W}$ is a symmetric PSD matrix as the addition of two likewise matrices. Instead of resorting to matrix inversion, we compute V using a linear solver for symmetric PSD matrices. To compute the necessary gradients for backpropagation

of gradients through Equation (3.1), we rely on matrix calculus and use of three Kronecker product $\otimes$ properties [189]: 1) $\text{vec}(\mathbf{QWE}) = (\mathbf{E}^\mathsf{T} \otimes \mathbf{Q})\text{vec}(\mathbf{W})$, 2) $T_{m,n}\text{vec}(Q) = \text{vec}(\mathbf{Q}^\mathsf{T})$ and 3) $(\mathbf{Q} \otimes \mathbf{W})(\mathbf{E} \otimes \mathbf{R}) = (\mathbf{QE} \otimes \mathbf{WR})$. The gradients of any linear solver for symmetric matrices are the following

$$\begin{aligned}\frac{\partial g(\mathbf{V})}{\partial \mathbf{b}} &= \frac{\partial \mathbf{V}}{\partial \mathbf{b}}\frac{\partial g(\mathbf{V})}{\partial \mathbf{V}} = \frac{\partial \mathbf{W}^{-1}\mathbf{b}}{\partial \mathbf{b}}\frac{\partial g(\mathbf{V})}{\partial \mathbf{V}} \\ &= \mathbf{W}^{-\mathsf{T}}\frac{\partial g(\mathbf{V})}{\partial \mathbf{V}} = \mathbf{W}^{-1}\frac{\partial g(\mathbf{V})}{\partial \mathbf{V}}\end{aligned} \tag{3.3}$$

$$\begin{aligned}\frac{\partial g(\mathbf{V})}{\partial \,\text{vec}(\mathbf{W})} &= \frac{\partial \mathbf{V}}{\partial \,\text{vec}(\mathbf{W})}\frac{\partial g(\mathbf{V})}{\partial \mathbf{V}} = \frac{\mathbf{W}^{-1}\mathbf{b}}{\partial \,\text{vec}(\mathbf{W})}\frac{\partial g(\mathbf{V})}{\partial \mathbf{V}} \\ &= \frac{\partial \,\text{vec}(\mathbf{W}^{-1})}{\partial \,\text{vec}(\mathbf{W})}\frac{\partial \mathbf{W}^{-1}\mathbf{b}}{\partial \,\text{vec}(\mathbf{W}^{-1})}\frac{\partial g(\mathbf{V})}{\partial \mathbf{V}} \\ &= \frac{\partial \,\text{vec}(\mathbf{W}^{-1})}{\partial \,\text{vec}(\mathbf{W})}\frac{\partial \mathbf{W}^{-1}\mathbf{b}}{\partial \,\text{vec}(\mathbf{W}^{-1})}\frac{\partial g(\mathbf{V})}{\partial \mathbf{V}} \\ &= \frac{\partial \,\text{vec}(\mathbf{W}^{-1})}{\partial \,\text{vec}(\mathbf{W})}\frac{\partial (\mathbf{b}^\mathsf{T} \otimes \mathbf{I})\,\text{vec}(\mathbf{W}^{-1})}{\partial \,\text{vec}(\mathbf{W}^{-1})}\frac{\partial g(\mathbf{V})}{\partial \mathbf{V}} \\ &= (-\mathbf{W}^{-1} \otimes \mathbf{W}^{-1})^\mathsf{T}(\mathbf{b}^\mathsf{T} \otimes \mathbf{I})^\mathsf{T}\frac{\partial g(\mathbf{V})}{\partial \mathbf{V}} \\ &= (-\mathbf{W}^{-1} \otimes \mathbf{W}^{-1})(\mathbf{b} \otimes \mathbf{I})\frac{\partial g(\mathbf{V})}{\partial \mathbf{V}} \\ &= -(\mathbf{V} \otimes \mathbf{W}^{-1})\frac{\partial g(\mathbf{V})}{\partial \mathbf{V}} = -\mathbf{V} \otimes \frac{\partial g(\mathbf{V})}{\partial \mathbf{b}}\end{aligned} \tag{3.4}$$

As such,

$$\frac{\partial g(\mathbf{V})}{\partial \mathbf{W}} = -\frac{\partial g(\mathbf{V})}{\partial \mathbf{b}}\mathbf{V}^\mathsf{T} \tag{3.5}$$

After calculating the gradients of the linear solver, the final step is the computation of the gradients with respect to the handle position regression $\tilde{\mathbf{H}}$ and learnable dependency matrix $\mathbf{A}$. The gradient of the first quantity is straight forward to calculate using Equation (3.3).

$$\begin{aligned}\frac{\partial g(\mathbf{V})}{\partial \tilde{\mathbf{H}}} &= \frac{\partial \mathbf{b}}{\partial \tilde{\mathbf{H}}}\frac{\partial g(\mathbf{V})}{\partial \mathbf{b}} = \frac{\partial (\mathbf{L}^\mathsf{T}\mathbf{L}\mathbf{T} + \mathbf{A}^\mathsf{T}\tilde{\mathbf{H}})}{\partial \tilde{\mathbf{H}}}\frac{\partial g(\mathbf{V})}{\partial \mathbf{b}} \\ &= \mathbf{A}\frac{\partial g(\mathbf{V})}{\partial \mathbf{b}}\end{aligned} \tag{3.6}$$

Lastly we provide the gradient with respect to the learnable dependency matrix $\mathbf{A}$

$$\frac{\partial g(\mathbf{V})}{\partial \operatorname{vec}(\mathbf{A})} = \frac{\partial \mathbf{b}}{\partial \operatorname{vec}(\mathbf{A})}\frac{\partial g(\mathbf{V})}{\partial \mathbf{b}} + \frac{\partial \operatorname{vec}(\mathbf{W})}{\partial \operatorname{vec}(\mathbf{A})}\frac{\partial g(\mathbf{V})}{\partial \operatorname{vec}(\mathbf{W})} \tag{3.7}$$

$$\begin{aligned}\frac{\partial \mathbf{b}}{\partial \operatorname{vec}(\mathbf{A})} &= \frac{\partial(\mathbf{L}^\mathsf{T}\mathbf{L}\mathbf{T} + \mathbf{A}^\mathsf{T}\tilde{\mathbf{H}})}{\partial \operatorname{vec}(\mathbf{A})} = \frac{\partial \operatorname{vec}(\mathbf{A}^\mathsf{T}\tilde{\mathbf{H}})}{\partial \operatorname{vec}(\mathbf{A})} \\ &= \frac{\partial \operatorname{vec}(\mathbf{A}^\mathsf{T})}{\partial \operatorname{vec}(\mathbf{A})}\frac{\partial(\tilde{\mathbf{H}} \otimes \mathbf{I})\operatorname{vec}(\mathbf{A}^\mathsf{T})}{\partial \operatorname{vec}(\mathbf{A}^\mathsf{T})} \\ &= \frac{\partial \mathbf{T}_{K,N}\operatorname{vec}(\mathbf{A})}{\partial \operatorname{vec}(\mathbf{A})}(\tilde{\mathbf{H}}^\mathsf{T} \otimes \mathbf{I})^\mathsf{T} = \mathbf{T}_{N,K}(\tilde{\mathbf{H}} \otimes \mathbf{I})\end{aligned} \tag{3.8}$$

$$\begin{aligned}\frac{\partial \operatorname{vec}(\mathbf{W})}{\partial \operatorname{vec}(\mathbf{A})} &= \frac{\partial(\mathbf{L}^\mathsf{T}\mathbf{L} + \mathbf{A}^\mathsf{T}\mathbf{A})}{\partial \operatorname{vec}(\mathbf{A})} = \frac{\partial \operatorname{vec}(\mathbf{A}^\mathsf{T}\mathbf{A})}{\partial \operatorname{vec}(\mathbf{A})} \\ &= \mathbf{I}_N \otimes \mathbf{A}^\mathsf{T} + (\mathbf{A}^T \otimes \mathbf{I}_N)\mathbf{T}_{K,N}\end{aligned} \tag{3.9}$$

In practice we initialize the dependency matrix $\mathbf{A}$ based on Farthest Point Sampling (FPS) [190] of the mesh, shortlisting a set of vertices $\{v_k\}, k = 1 \dots K$ that are approximately equidistant. For each vertex v_k we initialize the $k-$ th row of $\mathbf{A}$ based on the geodesic distance of the vertices to v_k:

$$\mathbf{A}[i,k] = \frac{\exp(1/d_{i,v_k})}{\sum_j \exp(1/d_{j,v_k})} \tag{3.10}$$

We note that at test time we have a constant affinity matrix, $\mathbf{A}$. Combined with the fixed values of $\mathbf{L}$ and $\mathbf{T}$, we can fold the solution of the linear system in Eq. 3.2 into a linear layer:

$$\mathbf{V}^* = \mathbf{C} + \mathbf{D}\tilde{\mathbf{H}}, \tag{3.11}$$

with $\mathbf{C}$ and $\mathbf{D}$ being constant matrices obtained by multiplying both sides of Eq. 3.2 by the inverse of $\mathbf{L}^\mathsf{T}\mathbf{L} + \mathbf{A}^\mathsf{T}\mathbf{A}$. We can thus interpret our method as using at training time template-driven regularization to solve the ill-posed problem of monocular 3D reconstruction, but being as simple and fast as a linear layer at test-time. In Figure 3.2 we visualized the rows of matrix $\mathbf{A}$ for a trained network on videos of

horses. An interesting observation is the locality of the learned distributions capturing intuitive areas of the template and allowing for accurate mesh deformations.

3.3.2 Motion-based 3D supervision

Having described our deformation model, we turn to the use of video information for network training. We rely on optical flow [184] to deliver pixel-level correspondences between consecutive object-centered crops. Unlike traditional 3D vision which relies on category-agnostic point trajectories for 3D lifting, e.g. through factorization [165], we use the flow-based correspondences to constrain the mesh-level predictions of our network in consecutive frames.

In particular, our network takes as input a frame at time t and estimates a mesh $\mathbf{V}_t$ and a weak perspective camera $\mathbf{C}_t$. A mesh vertex i that is visible in both frames t and $t+1$ will project to two image points $\mathbf{p}_{i,t} = \pi(\mathbf{V}_{i,t}, \mathbf{C}_t)$ and $\mathbf{p}_{i,t+1} = \pi(\mathbf{V}_{i,t+1}, \mathbf{C}_{t+1})$ where π amounts to weak perspective projection. As such the displacement of point $\mathbf{p}_{i,t}$ according to our network will be $\tilde{\mathbf{u}}_i = \mathbf{p}_{i,t+1} - \mathbf{p}_{i,t}$.

This prediction is compared to the optical flow value u_i delivered at $\mathbf{p}_{i,t}$ by a pretrained network [184] that we treat as the ground-truth. We limit our supervision to image positions in the interior to the object masks and vertices visible in both frames; vertex visibility is recovered by z-buffering, available in any differentiable renderer. We denote the vertices that are eligible for supervision in terms of a binary visibility mask $\gamma : \{1, \ldots, \Gamma\} \rightarrow \{0, 1\}$.

We combine these terms in a 'motion re-projection' loss expressed as follows:

$$L_{\text{motion}} = \frac{1}{\sum_{i=1}^{\Gamma} \gamma_i} \sum_{i=1}^{\Gamma} \gamma_i \left\| \mathbf{u}_i - \tilde{\mathbf{u}}_i \right\|_1 \tag{3.12}$$

where we use the ℓ_1 distance between the flow vectors for robustness and average over the number of visible vertices to avoid pose-specific value fluctuations. Since $\tilde{\mathbf{u}}_i = \pi(\mathbf{V}_{i,t+1}, \mathbf{C}_{t+1}) - \pi(\mathbf{V}_{i,t}, \mathbf{C}_t)$ continuously depends on the camera and mesh predictions of our network, we see that this loss can be used to supervise both the camera and mesh regression tasks.

This loss obviously penalizes the cases where limb articulation observed in the

image domain is not reflected in the 3D reconstructions, effectively forcing the 3D reconstructions to become more 'agile' by deforming the mesh more actively. Interestingly, we observed that beyond this expected behaviour this loss has an equally important effect on the camera prediction, by forcing the backprojected mesh to 'stand still' in the interior of objects: even though different camera poses could potentially backproject to the same object in a single image, a change in the camera across frames will cause large 2D displacements for the corresponding 3D vertices. Beyond motion loss we further make use of losses capturing keypoint, pixel-level appearance and boundary level supervision for the shape.

Keypoint reprojection loss, as in [4], penalizes the ℓ_1 distance between surface-based predictions and ground truth keypoints, when available:

$$L_{\text{kp}} = \sum_i \|\mathbf{k}_i - \pi(\mathbf{K}_i \mathbf{V}, \mathbf{C})\|_1 ,$$

where $\mathbf{K}_i$ is a fixed vector that regresses the $i-$th semantic keypoint in 3D from the 3D mesh.

Texture Loss compares the mesh-based texture and the image appearance in terms of the perceptual similarity metric of [220] after masking by the silhoutette S:

$$L_{\text{pixel}} = \text{dist}\left(\tilde{I} \odot S, I \odot S\right).$$

As in [3] we enforce symmetric texture predictions by using a bilateral symmetric viewpoint.

Local Rigidity Loss, as in [54] aims at preserving the Euclidean distances between vertices in the extended neighborhood $\mathcal{N}(\mathbf{u})$ of a point $\mathbf{u}$:

$$L_{\text{rigid}} = \mathop{\mathbb{E}}_{\mathbf{u} \in V} \mathop{\mathbb{E}}_{\mathbf{u}' \in \mathcal{N}(\mathbf{u})} \left| \left\| V(\mathbf{u}) - V\left(\mathbf{u}'\right) \right\| - \left\| \bar{V}(\mathbf{u}) - \bar{V}\left(\mathbf{u}'\right) \right\| \right|$$

Region similarity loss compares the object support computed from the mesh by a differentiable renderer [193] to instance segmentations S provided either by

manual annotations or pretrained CNNs using their absolute distance:

$$L_{\text{mask}} = \sum_i \|S_i - f_{\text{render}}(V_i, \pi_i)\|$$

$$L_{\text{boundary}} = \mathop{\mathbb{E}}_{\mathbf{u}\in V} \mathscr{C}_{fg}(\pi(\mathbf{u})) + \mathop{\mathbb{E}}_{\mathbf{b}\in\mathscr{B}_{fg}} \min_{\mathbf{u}\in V} \|\pi(\mathbf{u}) - \mathbf{b}\|_2^2,$$

where as in [171, 54] the first term penalizes points of the predicted shape that project outside of the foreground mask using the Chamfer distance to it while the second term penalizes mask under-coverage by ensuring every point on the silhouette boundary has a mesh vertex projecting close to it.

3.3.3 Optimization-based learning and refinement

The objective function for our 3D reconstruction task combines motion supervision with other common losses in a joint objective function:

$$L_{\text{total}} = L_{\text{motion}} + L_{\text{kp}} + L_{\text{pixel}} + L_{\text{rigid}} + L_{\text{mask}} + L_{\text{boundary}}\,, \tag{3.13}$$

capturing keypoint, pixel-level appearance, rigidity priors, as well as mask- and boundary- level supervision for the shape.

In principle a neural network could successfully minimize the sum of these losses and learn the correct 3D reconstruction of the scene. In practice there are too many local minima in neural network optimization, which is further exacerbated in our weakly-supervised setting, where we are effectively requesting the network to both recover and learn the solution to an ill-posed problem for multiple training samples at the same time.

This has been observed even in human pose estimation [159, 33, 163, 191], where careful per-sample numerical optimization was shown to yield substantial performance improvements. Given that in our case we do not know the shape prior or have access to mocap recordings for supervision, it makes per-sample numerical optimization even more critical.

In particular we use focused, per-sample numerical optimization to refine the

Method	mIoU	PCK
CMR [3]	0.703	81.2
CSM [4]	0.622	68.5
A-CSM [2]	0.705	72.4
Ours		
8	0.64	84.6
16	0.676	89.8
32	0.688	89.7
64	**0.711**	**91.5**

Table 3.1: Ablation of deformation layer on CUB: Even with only 8 points, our handle-based approach outperforms all competing methods in terms of PCK, while with more handles both the mIoU and PCK scores improve further.

network's 'bottom-up' predictions so as to better match the image evidence by minimizing L_{total} with respect to the per-frame handles and camera poses; if the object were rigid this would amount to bundle adjustment, but in our case we also allow the handles to deform per frame. Our approach also applies to both videos and individual frames, where in the latter case we omit the motion-based loss. At test-time, as in the 'synergistic refinement' approach of [159], once the network has delivered its prediction for a test sample (frame/video), we start a numerical 'top-down' refinement of its estimate by minimizing L_{total} using masks delivered by an instance segmentation network and flow computed from the video if applicable. The approach comes with a computational overhead due to the need for forward-backward passes over the differentiable renderer for every gradient computation (we use Adam [192] for 50 iterations).

3.4 Experimental Results

3.4.1 Model architecture

We use the same encoder-decoder architecture that is presented in [3, 59]. Every image is encoded using an ImageNet pre-trained ResNet18 to a latent feature map $z \in \mathbf{R}^{4\times4\times256}$. A flattened version of z is processed with two MLP linear layers with output channels equal to 200 and the final result is given to the handle deformation predictor and camera predictor branches. The handle deformation branch provides the handle offsets $\Delta_H \in \mathbf{R}^{K\times3}$ and the camera predictor predicts the scale, translation

and rotation, which is encoded as quartenions, through 2 fully connected layers each with 200 channels. Finally, we use the same texture predictor architecture as [59] which takes as input the encoded features z and outputs the UV texture map $I^{uv} \in \mathbf{R}^{128\times256\times3}$. We also use Pytorch3D [193] as the differentiable renderer.

3.4.2 Per-sample optimization training framework

We build on the camera multiplex training procedure proposed by Goel et al. [59] to train the proposed method. We provide a small review of the method for coherency, however we refer the reader to [59] for a thorough explanation and technical details.

The authors propose using a learnable set of possible camera hypotheses for each training instance that is learned simultaneously with the rest of the 3D reconstruction CNN. In detail, each training instance i has $C_i = \{\pi_1, \ldots, \pi_{N_c}\}$ associated camera hypotheses, which are modelled as weak perspective cameras $(s \in \mathbf{R}, \mathbf{t} \in \mathbf{R}^2, \mathbf{q} \in \mathbf{R}^4)$, that are retrieved from a 'camera database' during training using unique indices for each training sample. Each camera π_i is optimized to minimize the reconstruction losses for the silhouette $L_{sil,i}$ and the texture $L_{tex,i}$ like those used in our proposed framework. In the case of the motion loss $L_{motion,i}$, we avoid using the cartesian product between the cameras pools C_i and C_{i+1} since that would require excessive computational resources. Instead, we form pairs for each of the cameras in the hypothesis pool C_i with the nearest camera in the pool C_{i+1}. For updating the parameters of the method, the resulting losses $L_i = L_{sil,i} + L_{tex,i} + L_{motion,i}$ of the camera set are used as a distribution over the most likely camera pose. This is encoded as a probability $p_i = \frac{e^{-L_i}}{\sum_j e^{-L_j}}$ for π_i to be the most likely camera. Using the computed distribution the final loss is formulated as

$$L = \sum_i p_i \left(L_{\text{sil},i} + L_{\text{tex},i}\right).$$

The final step of this training procedure is to train a camera predictor using the most probable camera of each training image conditioned on the image features extracted from the backbone CNN that is driving the whole reconstruction process.

Building on this optimization driven paradigm, we extend the training protocol

with three distinct modifications. First of all, alongside the silhouette and textures losses we incorporate also the motion re-projection loss that is described in detail in the main paper. Furthermore, we extend the camera set with per image deformations D which is only one per image unlike the multiple cameras. In our pipeline, each training image i has a camera set C_i and a single handle deformation vector $D_i \in \mathbf{R}^{K\times 3}$ (with K being the number of handles) that are both used to express a multi-hypotheses distribution similar to [59]. Thirdly, unlike the aforementioned work, we simultaneously train our deformation and camera prediction branches using the most probable explanation for both the camera and deformation in accordance to the resulting silhouette and texture losses. This is achieved by adding two extra losses in the total loss function that minimize the ℓ_2 norm of the difference between the predicted quantity and the optimized one retrieved from the 'database' for each of the cameras and deformations.

In the case of keypoint trained networks, we set $N_c = 1$ and initialize the camera with a rigid SfM camera in accordance to CMR [3]. For all other experiments, we use $N_c = 8$ and initialize the camera set C for every image in the training set with camera hypotheses whose azimuth is uniformly spaced on the viewing sphere. The handle deformations D are initialized with zeros which corresponds to the template shape. As a first step, each camera is optimized using the silhouette loss L_{sil} and motion loss L_{motion} using the template shape before training the rest of the method. We implement a drop hypotheses procedure [59] to reduce the computational complexity where the most improbable hypotheses are discarded from the camera set. In detail, after 20 epochs we keep the four most probable cameras and after 100 epochs we keep only the 2 most probable cameras. Any training augmentations that scale and translate the training image i are directly encoded as affine transformations on the respective camera set C_i while the deformation D_i remains unchanged since the depicted deformation of the object remains identical.

3.4.3 Data

We report quantitative reconstruction results for objects with keypoint-annotated datasets, i.e birds, horses, tigers and cows. For a wide set of objects a dataset is

Table 3.2: Keypoint Reprojection Accuracy We report PCK accuracy (higher is better) achieved by the methods [4, 2] for three different objects. We also indicate datasets used to train each method alongside with their source of supervision.

Method	Supervision			Training Dataset	Horse		Cow	Tiger
	KP	Mask	Motion		TigDog	Pascal	Pascal	TigDog
CSM [4]	✓	✓		*P + I*	59.0	46.4	52.6	-
ACSM [2]	✓	✓		*P + I*	57.8	57.3	56.8	-
ACSM [2]	✓	✓		*TD*	68.7	44.4	-	36.2
Ours, inference	✓	✓	✓	*TD*	74.7	57.2	-	51.9
Ours, with refinement	✓	✓	✓	*TD*	**83.1**	**69.5**	-	**55.7**
CSM [4]		✓		*P + I*	44.7	49.7	37.4	-
ACSM [2]		✓		*P + I*	58.1	54.2	43.8	-
ACSM [2]		✓		*TD + YV*	26.7	33.3	-	15.1
Ours, inference		✓	✓	*TD + YV*	42.5	31.6	44.6	28.4
Ours, with refinement		✓	✓	*TD + YV*	**61.3**	**54.9**	**53.9**	**32.5**

Datasets: Pascal (P), ImageNet (I), TigDog (TD), YVIS (YV)

collected, mainly from available video datasets [194, 195]. All of the videos in our datasets have been filtered manually for occluded or heavily truncated clips that are removed from the dataset.

Birds We use the CUB [5] dataset for training and testing on birds which contains 6000 images. The train/val/test split we use for training and report is that of [3]. While this dataset is single-frame, we use it to compare our deformation module with prior works on similar grounds.

Quadrupeds (Horses, Tigers) We use the TigDog Dataset [194] which contains keypoint-annotated videos of horses and tigers. The segmentation masks are approximate since they are extracted using MaskRCNN [196]. We also drop the neck keypoint for both categories since there is a left-right ambiguity in all annotations. For every class we keep 14 videos purely for evaluation purposes and train with the rest, i.e 53 videos for horses and 44 for tigers. For these classes, the number of handles is set to $K = 16$.

Quadrupeds (giraffe, zebras and others) We use Youtube Video Instance Segmentation dataset (YVIS) [195], that contains videos for a wide variety of objects, to 3D reconstruct more animal classes. The cow category is used for evaluation against other methods and for the rest of the classes we provide qualitative

Table 3.3: Ablation of deformation layer on CUB: Even when using only 8 points, our handle-based approach outperforms all competing methods in terms of PCK, while with more handles both the mIoU and PCK scores further improve.

Method	mIoU	PCK
CMR [3]	0.703	81.2
CSM [4]	0.622	68.5
A-CSM [2]	0.705	72.4
Ours:		
8	0.64	84.6
16	0.676	89.8
32	0.688	89.7
64	**0.711**	**91.5**

results due to the lack of keypoint-annotated data.

For all categories we downloaded template shapes from the internet and down-sampled to a fixed number of $N = 642$ vertices. For evaluation we use identical template shape and keypoint annotations to those of [2] for all classes.

3.4.4 Results

Table 3.4: Ablation of motion and optimization based reconstruction for horses and tigers classes.

	Horse				Tiger			
	w/ L_{Motion}		w/o L_{Motion}		w/ L_{Motion}		w/o L_{Motion}	
	mIoU	PCK	mIoU	PCK	mIoU	PCK	mIoU	PCK
Inference	0.536	74.7	0.519	71.5	0.538	51.9	0.52	49.0
Mask	0.691	79.5	0.691	79.5	0.663	51.8	0.616	49.4
Mask and motion	0.631	83.1	0.675	72.5	0.76	55.7	0.64	54.0

3.4.4.1 Handle-based deformation evaluation

We start with the CUB [5] dataset where we use the exact supervision of A-CSM [4]. We outperform the state-of-the-art system on reconstruction [3] by a significant margin in both mean Intersection over Union (mIoU) and keypoint reprojection accuracy (PCK). PCK is a measure of the accuracy of 3D to 2D correspondences. It is computed by first defining a threshold distance d, then counting the number of keypoints (or correspondences) for which the distance between the predicted 2D

keypoint and the true 2D keypoint is less than d. All methods PCK scores were computed at normalized distance threshold 0.1 with respect to image size. PCK and mIoU (mean Intersection over Union) are 2D metrics that measure the accuracy of 2D predictions, such as keypoints or object segmentation, but do not directly measure the quality of the underlying 3D predictions. These 2D losses can give an indication of how well the 3D predictions align with the 2D image plane, but they do not provide information about the accuracy of the 3D geometry or the object's pose and location in 3D space. We ablate in particular the effect of the number of handles on the achieved 3D reconstruction in Table 3.3. We observe that our results are outperforming previous methods even with a very small number of handles, however increasing the number of handles allows for improved performance. Compared to CMR, we use handle-based deformation as opposed to per point deformation prediction for improved control and optimization. Handle-based deformation allows for coherent 3D surface areas to be grouped together, making it easier to optimize the overall solution. This results in a deformation method that is easier to optimize and in return surpasses the performance of CMR even with as little as 8 handles. We also provide qualitative results in Figure 3.3 where we show that our method is capable of correctly deforming the template mesh to produce highly flexible wings, while the alternative methods barely capture open wing variation. These results clearly indicate the merit of our handle-based deformation layer.

3.4.4.2 Motion- and Optimization- based evaluation

In Table 3.4, we perform an extensive ablation of the impact of our motion-based supervision, and optimization-based reconstruction for the category of horses. We consider firstly the impact that motion-based supervision has as a source of training (left versus right columns). We observe that motion supervision systematically improves accuracy across all configurations and evaluation measures.

When optimizing at test time as post-processing we observe how the terms that drive the optimization influence the final results: when using only masks we have a marked increase in mIoU, and a smaller increase in PCK, while when taking motion-based terms into account as well the increase in mIoU is not as big but we

Figure 3.1: Quadruped reconstructions of our proposed method. We provide renderings of the 3D reconstruction using the estimated camera pose, a different viewpoint and the texture reconstruction.

attain the highest improvement in PCK. The same pattern is observed for the Tiger category in the smaller ablation Table 3.4. We visualize in Figure 3.4 the mean shape of several classes along with the first 3 common deformation modes. These visualizations are created by running PCA on the collection of predicted handle offsets for the test set. We visualize the first 3 principal components while modulating the weight for each from -3 to 3.

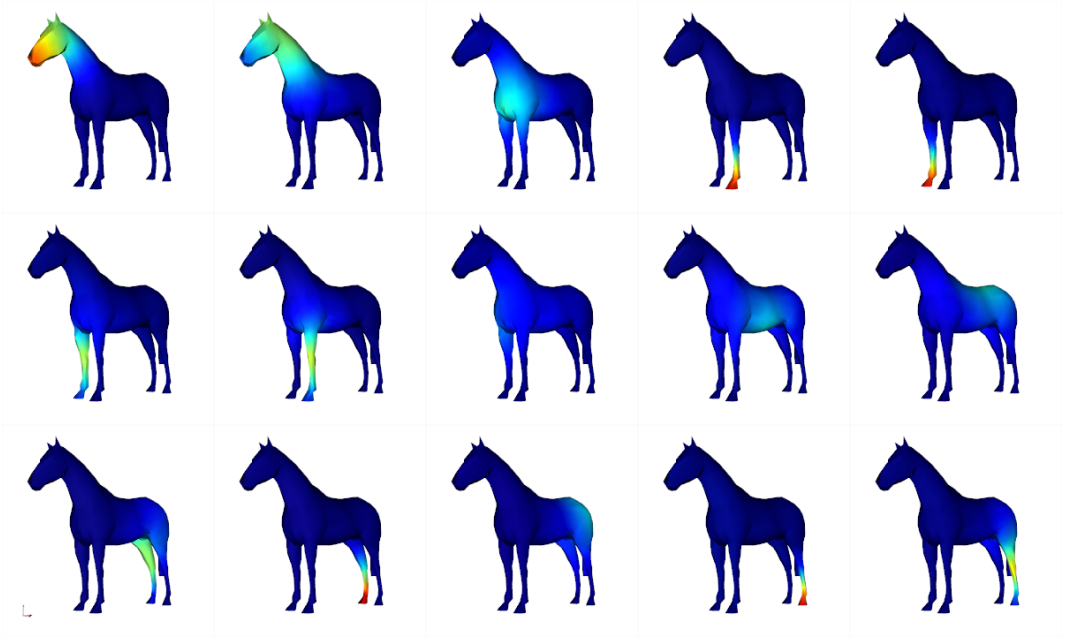


Figure 3.2: Handle Influence: We visualize each row of the matrix **D** for the horse class. We observe that the associations between the learned handles and the 3D points are localized to areas of intuitive interest such as legs, tail and the head which allow for accurate deformation of the template.

3.4.4.3 Comparisons on more categories

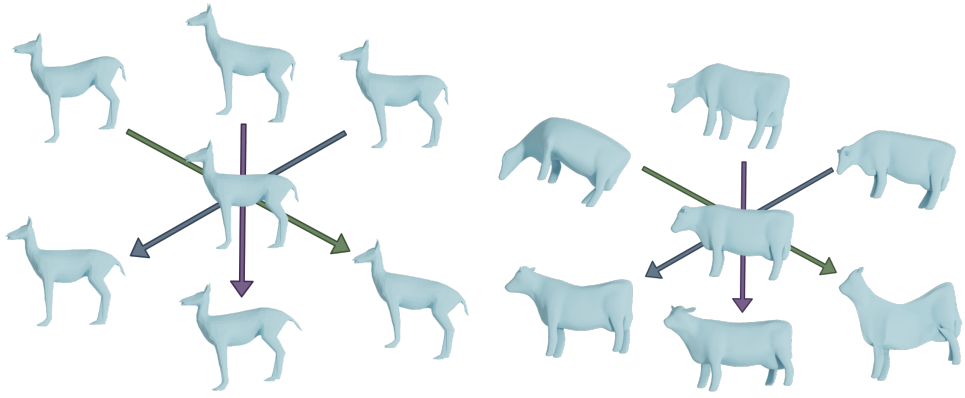
In Table 3.2 we report results on more categories where we have been able to compare to the currently leading approaches to monocular 3D reconstruction [3, 4, 2]. We note that several of the datasets used in these works are not publicly available (e.g. Imagenet post-processed images for the relevant categories), as such our training data are not directly comparable. Still we note that we use a very small number of videos (53 for horses, 44 for tigers, 24 for cows) compared to the thousands of images available in Imagenet or the hunderds in Pascal used by the existing approaches.

Starting with the comparison on horses for the case where keypoints are available, we observe that our inference-only method has a clear lead when testing on the TigDog dataset (the other methods have not been trained on TigDog), while optimization results in a further boost. When tested on Pascal (our system was not trained on Pascal nor ImageNet), our inference-only results are comparable to the best, while optimization gives us a clear edge. For cows we did not have videos

Figure 3.3: Bird Reconstructions: Qualitative comparisons between our method, CMR and ACSM with images from the CUB test set.

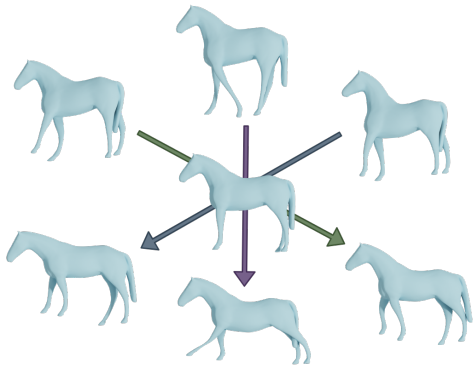
with cow keypoints, as such we did not train our approach on it, while for tigers we only report our own method's results since it has not been possible to train models for the existing methods.

Turning to results where we do not use keypoints, we observe that our method outperforms both CSM and ACSM when used in tandem with post-processing op-

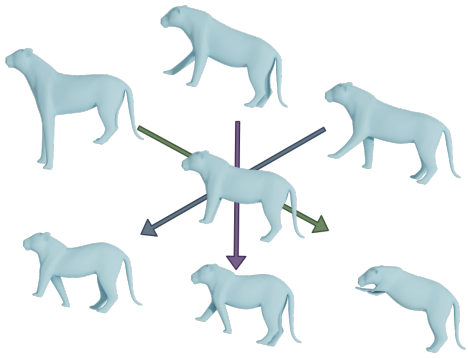


(a) Deer

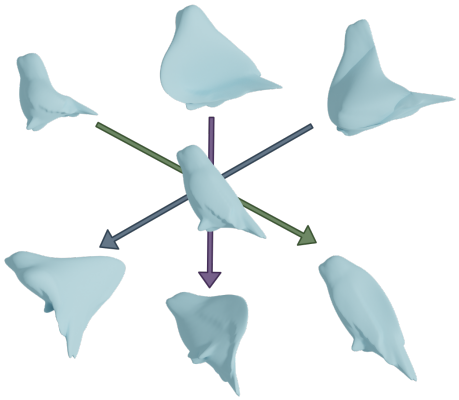
(b) Cow



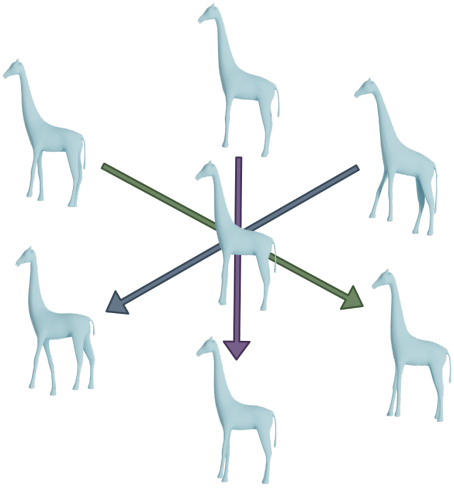
(c) Horse



(d) Tiger



(e) Birds



(f) Giraffe

Figure 3.4: Visualization of the predicted deformations for several objects by depicting the mean shape in the center and the first 3 modes obtained by PCA on the handle estimates obtained across the dataset.

timization, but overall we observe a larger drop in accuracy compared to the results obtained when keypoint supervision is available. For the case of cows we observe that even though our model was never trained on Pascal data, it outperforms the

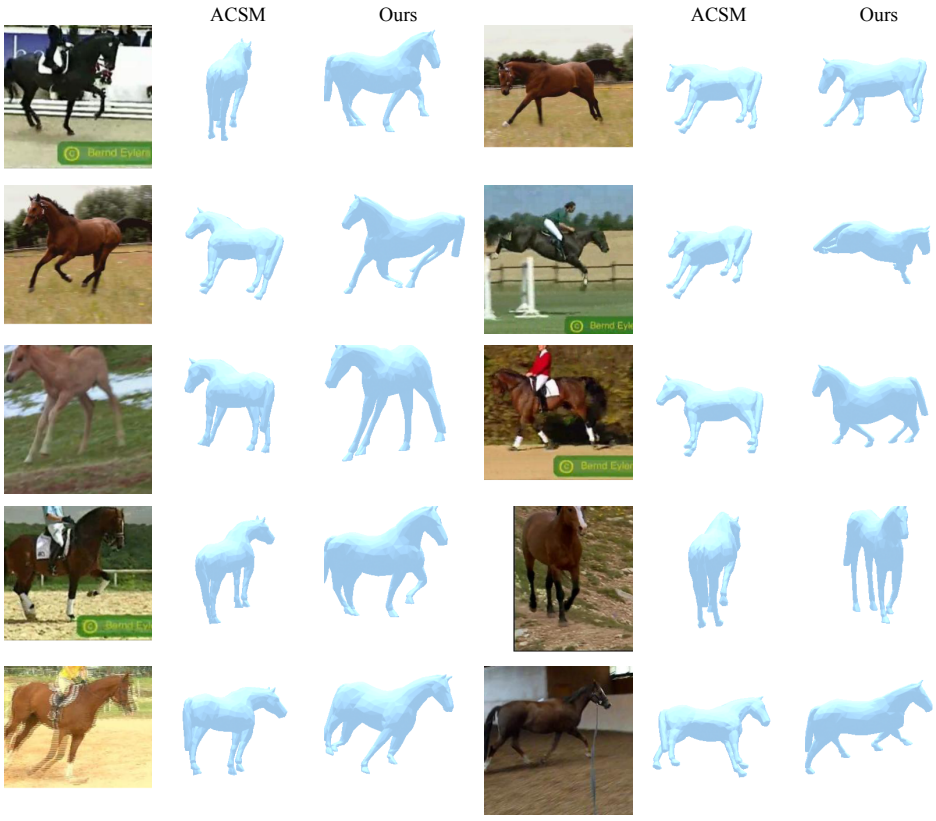


Figure 3.5: Reconstruction comparisons of our method and ACSM [2] for the object horse.

mask-supervised variants of both CSM and ACSM.

A pattern that is common for both sets of results is that post-processing optimization yields a substantial improvement in accuracy. As our qualitative results indicate in Figure 3.1, this is reflected also in the large amount of limb articulation achievable by our model. We provide a qualitative comparison between the proposed method and ACSM in Figure 3.5. The proposed method predicts more accurate camera and deformation reconstructions while requiring less human interventation in the way the meshes deform.

3.4.5 Failure Cases

We visualize some failure cases of the proposed method in Figure 3.6. Common failure cases are related to the inability to predict a good camera pose and the inference of simplistic textures. We believe that increasing the amount of training data will enables us to learn models capable of generalizing well to difficult instances like partially occlusions or close shots. Furthermore, a perspective camera would

Figure 3.6: Failure Cases: We visualize some failure modes of our method. The columns present the input image, 3D reconstruction from the predicted viewpoint and a different one and the predicted texture .

be better able to capture the close shots as in the elephant and fox on the second column. The current scaled orthographic camera used can not accurately capture the necessary depth dependent projection to the image space.

3.5 Conclusions

In this chapter, we presented a learning framework for monocular reconstruction that combines ideas from deep learning and geometry for the reconstruction of highly non-rigid objects while delivering interpretable and controllable deformation representations. We have derived the algorithmic formulation for differentiating through Laplacian solvers for mesh deformation. The end result is a deformation process that is interpretable and can be easily plugged in existing 3D software for post-processing. We have also extended approach with an optimization-driven post processing step that exploits image cues available to us during inference. Our experiments and reconstructions highlight the potential of the proposed method, showing that we can reconstruct challenging poses for a wide battery of highly deforming objects. We anticipate that the proposed framework will be useful for tasks such as graphics, AR, or robotic interaction with highly articulated animate object classes.

Chapter 4

Correspondence-driven monocular 3D category reconstruction

In this chapter we propose a method that uses self-supervision to 3D reconstruct highly non-rigid objects from images. The method To The Point (TTP) recovers a 3D shape from a 2D image by first regressing the 2D positions corresponding to the 3D template vertices and then jointly estimating a rigid camera transform and non-rigid template deformation that optimally explain the 2D positions through the 3D shape projection. By relying on 3D-2D correspondences we use a simple per-sample optimization problem to replace CNN-based regression of camera pose and non-rigid deformation and thereby obtain substantially more accurate 3D reconstructions. We treat this optimization as a differentiable layer and train the whole system in an end-to-end manner. Backpropagation through the optimization scheme is achieved via implicit differentiation. We report systematic quantitative improvements on multiple categories and provide qualitative results comprising diverse shape, pose and texture prediction examples.

The contents of this chapter were presented to the Conference on Neural Information Processing Systems (NeurIPS).

4.1 Introduction

Monocular 3D reconstruction of general categories is a task that humans perform with ease, yet remains challenging for computer vision due to its inherently ill-

posed nature: the observed 2D image is the result of a confluence of multiple sources of variation, including non-rigid intra-category shape variation, rigid transforms due to camera pose, as well as appearance variation. CNNs can easily learn to discard appearance variation, yet the treatment of the geometric sources of variability remains elusive. Even though strongly-supervised approaches have delivered compelling results e.g. for human reconstruction [197], for general categories we need to rely on weaker forms of supervision as well as self-supervision stemming from the know-how of computer vision.

3D vision has traditionally relied on correspondences to recover both rigid scenes from 2D images for the Structure-from-Motion (SFM) problem [166, 198, 199] as well as the more challenging problem of recovering Non-Rigid structure from 2D point tracks (NR-SFM) [200, 201, 169, 202]. In all those problems 3D reconstruction is accomplished by minimizing the reprojection error between the 3D positions of the inferred 3D scene and their 2D image correspondences. While these solutions have been developed for the (potentially deformable) single-instance case, the idea of relying on correspondences to supervise monocular 3D reconstruction has transpired in recent deep learning works.

CNN-driven monocular 3D category reconstruction [3, 59, 60, 61] has largely relied on self-supervision for 3D recovery expressed in terms of correspondence-based loss terms. For instance the geometric cycle loss terms of [62, 4, 2] are explicitly phrased in terms of correspondence established from UV maps while the texture-driven loss terms of [3, 62, 61, 63, 60] are implicitly relying on pixel correspondence. The common ground of such loss terms is that if the 3D shape is predicted correctly, it should project to the image in a way that is consistent with the 2D observations, as measured in a pixel-by-pixel sense. These correspondence terms are typically used in tandem with explicit geometric priors such as 3D symmetry [59, 3], predefined camera viewpoint ranges [4, 2, 59], or predefined object scales [4, 2, 59, 61] in order to tackle the ill-posed nature of the problem and the presence of multiple local minima in the associated learning problem.

Local minima however emerge even in the simpler single-instance case of NR-

SFM, while highly sophisticated optimization schemes have been introduced to address them, e.g. [203]. Current CNN-based approaches seem to ignore this problem and further exacerbate it by delegating the solution of 3D reconstruction to back-propagation with SGD: separate network heads are tasked with regressing the camera pose and non-rigid deformation given an image and are trained in an end-to-end manner, aiming to minimize the correspondence-driven losses. We argue that this is making optimization harder: network training aims at simultaneously establishing the association between images and rigid and non-rigid pose parameters as well as solving the 3D reconstruction problem in terms of these parameters. Each of these problems is hard enough in isolation and putting them together makes the optimization even harder.

This challenge is reflected in the complicated numerical schemes currently used to mitigate local minima; for instance [59, 61] use multiple camera hypotheses during both training and testing. The number of hypotheses can range from 8 to up to 40 for a single reconstruction and the hypotheses have to be accompanied with a probabilistic method to select the most accurate pose either predicted by an MLP [4] or using heuristic loss-based weighting schemes [59]. Another example of brittle optimization, even when keypoint supervision is available, are the works of [3, 204] where in a first stage SFM/NR-SFM is used to get the camera pose right based on keypoint supervision, which is then followed by optimization with image-based losses to recover a mesh. This challenge has been observed also in the strongly-supervised case of human pose estimation, and the use of per-sample numerical optimization [28] was shown to improve performance in [159, 33, 163, 191].

In this work we deviate from the current practice of using a CNN to regress camera and mesh deformation estimates. Instead, during both training and testing we solve a per-sample optimization problem that explicitly aims at providing a 3D reconstruction that projects “To The Point" (TTP). We take as input the 2D coordinates corresponding to the 3D vertices of a mesh and recover the 3D vertex positions by optimizing with respect to the rigid and non-rigid pose parameters through differentiable optimization [205, 206]. Hereafter, the term correspondence is used to

indicate the consistency or agreement between the 3D points and the 2D points predicted by a neural network. This refers to the projection of the 3D points onto a 2D image plane, and the consistency or accuracy of the neural network's prediction of these projections.

We obtain the 2D points required by our layer by only relying on mask annotations and optionally a small number of 2D semantic keypoint annotations, as well as self-supervision coming from the 3D reprojection loss. We jointly learn the 2D point regression and the 3D modes of shape variability through end-to-end optimization, while treating the per-instance rigid and non-rigid pose parameters as latent variables that are optimized on-the-fly, per sample.

We claim that predicting the correspondences is not only sufficient, but also more appropriate for driving monocular 3D reconstruction: it spares us from the use of any additional geometric priors and also yields state-of-the-art results while only relying on a single camera hypothesis. We evaluate our approach on 3D shape, pose and texture reconstruction on four objects categories using real-world datasets CUB [5] and PASCAL3D+ [207]. We demonstrate competitive 3D reconstruction quality to previous state-of-the-art methods and our ablation study confirms the importance of the self-supervised losses we employ.

4.2 Related Work

Monocular 3D reconstruction Recent works on this problem [54, 3, 4, 2, 63, 61, 57] have relied on varying forms of supervision. Earlier approaches [3, 54] treat the problem of 3D reconstruction from single images using known masks and manually labelled keypoints from single viewpoint image collections. Recent works [59, 63, 4] have removed the need for keypoints but introduced multiple viewpoint and deformation hypotheses accompanied with a probabilistic method to select the most accurate pose. [62, 208, 209] jointly recover cameras and non-rigid 3D meshes with single hypothesis-based networks, but limit themselves to simpler, almost planar categories like faces, or exploit symmetry priors, limiting their broader applicability.

Closer to our work is Canonical Surface Mapping (CSM) and Articulated Canonical Surface Mapping (ACSM) [4, 2] where the 3D representation is produced in the form of a rigid or articulated template using a 2D-to-3D cycle-consistency loss.

Non-rigid structure from motion (NR-SfM) The aim of NR-SfM is the recovery of the 3D shape and accompanying camera pose given only 2D landmarks without any explicit 3D supervision [169, 202, 170, 171]. Lately, several deep learning [62, 210, 211] methods have been proposed that surpassed the performance of traditional methods while being considerably faster. All of the aforementioned methods employ different priors to tackle the under-constrained problem of NR-SfM. The priors are embedded into the methods using low-rank subspaces [212, 169, 203], spatio-temporal domains [171, 178], equivariance constraints [210] or sparse basis coefficients using L1 constraints [211, 213, 214].

Learnable Optimization: Common methods for incorporating optimization as layers in deep neural networks include implicit function differentiation [205, 206, 215, 98, 216] and optimization unrolling [217, 218, 211]; we refer to [206, 205] for a survey. In 3D reconstruction recent works address the challenge of incorporating RANSAC in an end-to-end trainable pipeline for camera pose estimation based on the Perspective-n-Point (PnP) problem, such as differentiable blind PnP [98, 216] or DSAC [219]. Unlike these works, we do not have to address the combinatorial nature of correspondence, but rather focus on regressing the 2D image positions of a 3D template with a fixed number of vertices.

4.3 To-The-Point Monocular 3D Mesh Reconstruction

We start in Sec. 4.3.1 by introducing the 2D quantities predicted by our network, we then present our differentiable camera and mesh optimization layer in Sec. 4.3.2, and finally present the losses driving our end-to-end training in Sec. 4.3.4.

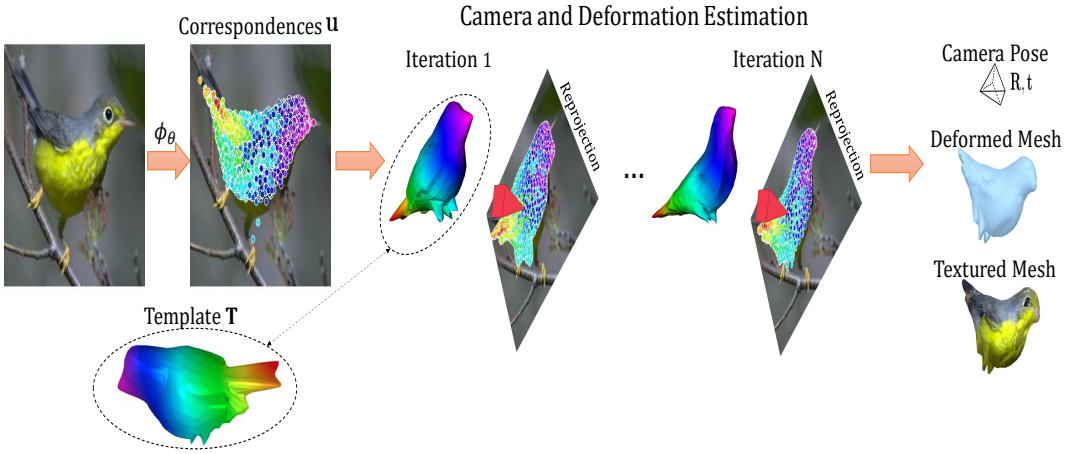


Figure 4.1: Overview of our method: Given an image we use a network ϕ_θ to regress the 2D positions $\mathbf{u}$ corresponding to the 3D vertices of a template; we then use a differentiable optimization method to compute the rigid (camera) and non-rigid (mesh) pose: in every iteration we refine our camera and mesh pose estimate to minimize the reprojection error between $\mathbf{u}$ and the reprojected mesh (visualized on top of the input image). The end result is the monocular 3D reconstruction of the observed object, comprising the object's deformed shape, camera pose and texture.

4.3.1 Predicting 3D to 2D Correspondences

Our method assumes that the template of our object category can be described in 3D in terms of N points. As shown in Fig. 4.1, given an image, we use a CNN, ϕ_θ, to regress the 2D coordinates $\mathbf{u} \in \mathbb{R}^{N\times 2}$ corresponding to these N 3D points. We also predict a visibility vector $\mathbf{v}$ where every $v_i \in [0,1]$ indicates whether the 2D to 3D correspondence is occluded in the image. Recognition of occluded points allows for accurate camera pose estimation by eliminating the influence of noisy predicted points belonging to the non-visible areas of the object.

4.3.2 Estimation of Pose and Deformation

Our aim is to estimate the camera pose and object deformation using only the predicted 2D points $\mathbf{u}$, the visibility vector $\mathbf{v}$ and the template mesh $\mathbf{T}$. We first introduce our assumptions about the rigid and non-rigid part of the shape and then turn to the resulting optimization problem.

Firstly, as in [3, 59, 4, 2, 63, 60], we model the 3D-to-2D projection through weak perspective [166]. This involves a 2×3 3D-to-2D "scaled orthographic" pro-

jection matrix of the following form:

$$\mathbf{C} = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \end{bmatrix}, \tag{4.1}$$

where the scaling factor s accounts for global scaling due to depth variation; given a set of 2D image points this is set to their standard deviation, yielding invariance of the pose parameters to similarity transforms of the 2D and 3D coordinate frames. The rigid pose parameters comprise a rotation $\mathbf{R}$ and translation $\mathbf{t}$ that account for viewing a 3D object $\mathbf{V}$ from a given camera position. The parametric estimates for the 2D projections of a 3D object can thus be obtained as follows:

$$\hat{\mathbf{u}}(\mathbf{R},\mathbf{t}) = \mathbf{C}(\mathbf{R}\mathbf{V}+\mathbf{t}) \tag{4.2}$$

where $\mathbf{V}$ is $3 \times N$, $\mathbf{u}$ is $2 \times N$, and we overload notation for $+$ assuming that $\mathbf{t}$ is replicated N times.

Having covered the rigid object case, we now turn to the modeling of non-rigid categories. For this we rely on the deformable template paradigm [6] commonly used also in the NR-SFM literature [169, 202], and obtain a shape estimate $\mathbf{V}$ by adding offsets $\mathbf{\Delta}_V = \mathbf{B}\mathbf{c}$ to a template shape $\mathbf{T}$, yielding $\mathbf{V} = \mathbf{\Delta}_V + \mathbf{T}$. Combined with Eq. 4.2, we have the following parametric estimate of the 2D positions:

$$\hat{\mathbf{u}}(\mathbf{C},\mathbf{R},\mathbf{t}) = \mathbf{C}(\mathbf{R}(\mathbf{T}+\mathbf{B}\mathbf{c})+\mathbf{t}) \tag{4.3}$$

which is bilinear in $\mathbf{R}$, $\mathbf{t}$ and $\mathbf{c}$.

So far we do not deviate substantially from recent works [3, 59, 63, 60] in terms of modelling: these also rely on the morphable model paradigm [6] and regress a shape update $\mathbf{\Delta}_V$ and a rotation matrix through network heads. A minor modelling difference is that we have a low-rank model for $\mathbf{B}\mathbf{c}$, while their shape update is driven by a high-dimensional latent vector. Our main difference is that rather than delegating to the network the task of predicting the 'right' values of $\mathbf{R},\mathbf{t},\mathbf{V}$, we directly optimize for them through a lightweight and differentiable optimization

scheme by exploiting our regressed 2D correspondences. This 'optimizes out' these parameters and ensures our 3D inference will project as accurately as possible to the 2D points, rather than delegating the optimization to backprop and the network heads.

Our "To-The-Point" approach aims at minimizing the following re-projection error between the predicted 2D points $\mathbf{u}$ and the 2D point estimates deliver by the parametric, 3D-based prediction:

$$l(\mathbf{R},\mathbf{t},\mathbf{c}) = \sum_{i=1}^{N} v_i \left\|\mathbf{u}_i - \hat{\mathbf{u}}_i(\mathbf{C},\mathbf{R},\mathbf{t})\right\|_2^2 + \gamma\|\mathbf{c}\|_2^2, \tag{4.4}$$

where we weigh the discrepancy between the two quantities by the regressed visibility, ensuring that the reconstruction process is robust to occluded points. We also add a regularization weight γ on the expansion coefficients to avoid instabilities in the first stages of training, when the basis $\mathbf{B}$ is still unknown and can lead to prematurely committing to large arbitrary deformations.

To minimize the loss term in Eq. (4.4), we use an alternating optimization scheme and perform separate updates for camera pose estimation and mesh deformation:

$$\hat{\mathbf{R}}^t, \hat{\mathbf{t}}^t = \arg\min_{\mathbf{R},\mathbf{t}} l(\mathbf{R},\mathbf{t},\hat{\mathbf{c}}^t), \text{subject to } \mathbf{R} \in SO(3) \tag{4.5}$$

$$\hat{\mathbf{c}}^{t+1} = \arg\min_{\mathbf{c}} l(\hat{\mathbf{R}}^t, \hat{\mathbf{t}}^t, \mathbf{c}) \tag{4.6}$$

where at each step we use some of the previously estimated quantities (denoted by hat) as fixed and optimize with respect with the remaining ones to update their estimates.

Starting from the optimization in Eq. 4.5, we satisfy the constraint that $\mathbf{R} \in SO(3)$ by using the angle-axis representation $\mathbf{r}$ of the rotation $\mathbf{R_r}$ such that $\mathbf{R_r} = \exp[\mathbf{r}]_\times$ where $[\cdot]_\times$ is the skew symmetric operator . The underlying non-linear problem is solved using the L-BFGS optimizer [79] and when $t > 1$ is initialized with the estimate of the previous iteration. To backpropagate through L-BFGS we

use the implicit function theorem as described in [98, 216].

We provide the closed form solution of the deformation step. The problem we solve is

$$\begin{aligned} l(\mathbf{c}) &= \sum_{i=1}^{N} v_i \left\| \mathbf{u}_i - \mathbf{C}(\mathbf{R}(\mathbf{T}_i + \mathbf{B}_i\mathbf{c}) + \mathbf{t}) \right\|_2^2 + \gamma \|\mathbf{c}\|_2^2 \\ &= \sum_{i=1}^{N} v_i \left\| \underbrace{(\mathbf{u}_i - \mathbf{C}\mathbf{R}\mathbf{T}_i - \mathbf{C}\mathbf{t})}_{\mathbf{y}_i} - \underbrace{\mathbf{C}\mathbf{R}^{t+1}\mathbf{B}_i}_{\boldsymbol{\omega}_i}\mathbf{c} \right\|_2^2 + \gamma \|\mathbf{c}\|_2^2 \end{aligned} \tag{4.7}$$

where $\mathbf{y} \in \mathbb{R}^2$ and $\boldsymbol{\omega} \in \mathbb{R}^{2\times K}$. The stationary point of (4.7) can be found by solving the following linear system:

$$\begin{aligned} \sum_{i=1}^{N} v_i(-2\boldsymbol{\omega}_i^\mathsf{T}\mathbf{y}_i + 2\boldsymbol{\omega}_i^\mathsf{T}\boldsymbol{\omega}_i\mathbf{c}) + \gamma\mathbf{I} = 0 \\ \sum_{i=1}^{N} (-2\boldsymbol{\omega}_i^\mathsf{T} v_i\mathbf{y}_i + 2\boldsymbol{\omega}_i^\mathsf{T} v_i\boldsymbol{\omega}_i\mathbf{c}) + \gamma\mathbf{I} = 0 \end{aligned} \tag{4.8}$$

The closed form solution can be further simplified using matrix representations. Assuming a lexicographic ordering of matrix $\boldsymbol{\omega}_i$ we formulate the matrix $\boldsymbol{\Omega}$ with size $2N \times K$. The matrix $\mathbf{X} \in \mathbb{R}^{2N\times 2N}$ is a diagonal matrix where each element of the diagonal corresponds to the visibility v_i. A lexicographic ordering is also applied to vectors $\mathbf{y}_i$ to retrieve the stacked vector $\boldsymbol{\Upsilon} \in \mathbb{R}^{2N}$. Using the formulated matrices the problem Eq 4.6 is solved with

$$\mathbf{c}^{t+1} = (\boldsymbol{\Omega}^\mathsf{T}\mathbf{X}\boldsymbol{\Omega} + \gamma\mathbf{I})^{-1}\boldsymbol{\Omega}^\mathsf{T}\mathbf{X}\boldsymbol{\Upsilon}. \tag{4.9}$$

Backpropagation through the matrix inversion of Eq. (4.9) is supported by all modern automatic differentiation frameworks, meaning that the chaining of the update steps can be treated as a differentiable layer and be used in tandem with end-to-end training.

Each row of $\boldsymbol{\Omega} \in \mathbb{R}^{2N\times K}$ is defined as $\boldsymbol{\Omega}_i = \mathbf{C}\mathbf{R}^{t+1}\mathbf{B}_i$ and $\mathbf{X} \in \mathbb{R}^{2N\times 2N}$ is a diagonal matrix containing the visibility vector $\mathbf{v}$. Finally, the vector $\boldsymbol{\Upsilon} \in \mathbb{R}^{2N}$ is

defined as $\Upsilon = \mathbf{u}_i - \mathbf{C}\mathbf{R}^{t+1}\mathbf{T}_i - \mathbf{C}\mathbf{t}^{t+1}$.

Even though the only quantities regressed by our CNN are the 2D positions and associated visibilities, the basis $\mathbf{B}$ that represents the shape variability of our category in Eq.4.3 is a parameter of this layer, is randomly initialized, and is estimated through back-propagation. As shown in Fig.4.5, the basis elements learned this way can be intuitively understood, while our experimental results indicate that they suffice for the accurate recovery of intricate mesh deformations.

4.3.3 Texture

The final part of monocular 3D reconstruction is the estimation of the texture of the reconstructed 3D shape. The texture of the object is sampled from the input image utilizing the predicted 2D points $\mathbf{u}$ closely resembling the sampling-based texturing method of CMR [3]. Unlike CMR, we do not predict uv locations to sample pixel values for the image with a dedicated learnable regressor, but use the predicted 2D points $\mathbf{u}$ that drive our whole 3D reconstruction process. For this we use a sampling-based texture approach where the face color is computed by interpolating the $\mathbf{u}$ coordinates and then sampling from the texture map, i.e the input image in our case. Any losses applied on the estimated texture back-propagate information to the predicted correspondences allowing us to use appearance image cues in addition to foreground masks to enable accurate correspondence predictions.

4.3.4 Loss terms

To train our approach we incorporate different losses focusing on pose estimation, texture prediction as well as mesh regularization.

Texture Loss compares the rendered textured image $\tilde{I}$ and the image appearance in terms of the perceptual similarity metric of [220] after masking by the silhouette S:

$$L_{\text{pixel}} = \text{dist}\left(\tilde{I} \odot S, I \odot S\right).$$

We also apply the loss on the symmetric texture predictions by using a bilateral symmetric viewpoint and average the two viewpoints. The soft symmetry constraint ensures that the texture of the non-visible side is still inline with the visible

side. This constraint has been employed in prior works [3, 63], however, unlike the proposed method it is commonly applied in a hard-coded manner by symmetrizing the texture across an axis.

Points Chamfer Distance enforces points to lie inside and cover the silhouette of the depicted object [54, 2]. In order to formulate our loss term we define C^{mask} as the Chamfer distance field of the binary mask of silhouette S. Silhouette consistency simply enforces the predicted 2D correspondences of an instance to lie inside its silhouette. This can be achieved by penalizing the points projected outside the instance mask by their distance from the silhouette. Silhouette coverage enforces the predicted points $\mathbf{u}_i$ to fully cover the mask of the depicted object and allows us to predict better camera poses and mesh deformations.

$$L_{\text{Chamfer}} = \underbrace{\sum_i C^{\text{mask}}(\mathbf{u}_i)}_{\text{silhouette consistency}} + \underbrace{\sum_{p \in S} \min_{\mathbf{u}_i} \|\mathbf{u}_i - \mathbf{p}\|_2}_{\text{silhouette coverage}}.$$

Region similarity loss compares the object support computed from the mesh by a differentiable renderer [193] to instance segmentations S provided either by manual annotations or pretrained CNNs using their absolute distance:

$$L_{\text{mask}} = \sum_i |S_i - f_{\text{render}}(V_i, \pi_i)|.$$

Cycle and Visibility loss Similarly to CSM [4] we use a cycle loss between the regressed 2D correspondences $\mathbf{u}$ and the projected 3D points to ensure that regressed points, that form neighborhoods in the template shape, remain close in the image space. The cycle loss is defined as $L_{\text{cycle}} = \sum_i \|\mathbf{u}_i - \pi(\mathbf{V})\|_2^2$. Furthemore, visibility of correspondences aids the camera pose estimation in weakly supervised cases. The visibility loss encourages the predictor ϕ_θ to encode the visible area of the mesh in an image by enforcing the predicted visibilities to be similar to those of the rendered z-buffer $\mathbf{v}^{\text{gt}}$

$$L_{\text{vis}} = \sum_i \left\| \mathbf{v}_i - \mathbf{v}_i^{\text{gt}} \right\|_1.$$

Equivariance Loss The point regressor should be robust to the pose variations. For each training image, we draw a random spatial transform $T_s(\cdot)$ from a predefined parameter range. We use random affine transformations (scale, rotation, and shifting) for spatial transforms as well as vertical flipping $T_v(\cdot)$. We pass both the input image I and transformed image $I' = T_s(I)$ through the ϕ_θ network and obtain the corresponding predictions $\mathbf{u}$ and $\mathbf{u}'$. For vertical flipping we retrieve two pose estimations $\mathbf{R}$ and $\mathbf{R}'$ for I and the flipped image $T_v(I)$. We compute the equivariance loss as follows:

$$L_{\text{eqv}} = \sum_i \left\| \mathbf{u}'_i - T_s(\mathbf{u}_i) \right\|_1 + \arccos \frac{1}{2} \left(\text{Tr}(T_v(\mathbf{R})\mathbf{R}') - 1 \right).$$

(Optional) Keypoint reprojection loss While we are primarily interested in training without any manual annotations, our approach can be extended to leverage an arbitrary number of high-level semantic keypoints. This is achieved by setting manually the 3D keypoints on the template mesh and encoding them as a matrix $\mathbf{K}$ acting on the mesh. The structure of $\mathbf{K}$ entails that each $\mathbf{K}_i$ is a fixed vector that regresses the $i-$th 3D semantic keypoint from the mesh. Given the 2D annotations for an image I and a camera π, a keypoint reprojection loss is formed between the groundtruth annotation and the projected 3D points:

$$L_{\text{kp}} = \sum_i \left\| \mathbf{k}_i - \pi \left(\mathbf{K}_i \mathbf{V} \right) \right\|_1.$$

As-rigid-as-possible (ARAP) constraint Without any mesh deformation regularization, the predicted mesh deformation will lead to arbitrary deformations exhibiting spikes and other anomalies. As such, we use the as-rigid-as-possible (ARAP) [68] constraint as a loss function similar to [60]. The predicted shape $\mathbf{V}$ is a locally rigid transformation from the predicted base shape $\mathbf{T}$ by:

$$L_{\text{arap}}\ (\mathbf{T}, \mathbf{V}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j \in N(i)} w_{ij} \left\| \left(\mathbf{V}^i - \mathbf{V}^j \right) - R_i \left(\mathbf{T}^i - \mathbf{T}^j \right) \right\|_2$$

where $N(i)$ represents the neighboring vertices of a vertex i, w_{ij} the cotangent

weights and R_i the best approximating rotation matrix, as described in [68]. Beyond mesh regularization, the same loss is applied on each basis component that leads to smooth and locally rigid components.

Even with ARAP there are cases where the network will squeeze the non-visible side of the reconstructed object. This erroneous deformation is not penalized by the ARAP loss, as long as it is locally rigid, and causes the method to predict flattened meshes. We further apply an l_2 constraint to the deformations to penalize the method to retain the original volume of the template.

4.4 Experiments

4.4.1 Datasets and Metrics

Datasets We present extensive ablation results and comparisons on bird reconstruction, as well as quantitative results on three more object categories (planes, cars, motorbikes). For birds we use the CUB [5] dataset for training and testing on birds which contains 6000 images. The train/val/test split we use for training and report is that of [3]. For the rest of the objects we use the Pascal3D+ dataset [207] and the associated pre-defined training and validation sets. Similarly to [3], we use both PASCAL VOC and Imagenet images to train our models and use Mask-RCNN [196] to obtain foreground masks for the ImageNet subset. For templates, we use identical to those of CSM [4] for CUB dataset and for PASCAL3D+ we select one of the available CAD models for each object.

Evaluation Metrics We evaluate our model on the CUB dataset [5] and report both the mean Intersection over Union (mIoU) and keypoint reprojection accuracy (PCK) following CMR [3].

For Pascal3D+ we report a canonical 3D mean Intersection over Union metric which measures the 3D overlap between the groundtruth and predicted deformed mesh; in order to compute the overlap, both meshes are voxelized using a 32 grid size before computing the 3D mIoU as in [59, 3, 221, 54].

Network Architecture Following prior work [3], we use a ResNet18 encoder to map an image I to a latent feature map $\mathbf{z} \in \mathbb{R}^{4\times4\times256}$. The position regressor is

Table 4.1: Evaluation of TTP performance on the CUB [5] dataset. We report mean and standard deviation (in parantheses, where applicable) of 2D mIoU and keypoint re-projection accuracy (PCK) along with related supervision signals for recent monocular 3D reconstruction methods.

	2D keypoints	Camera Priors	Camera Hypotheses	Rigid		Non-Rigid	
				mIoU ↑	PCK ↑	mIoU ↑	PCK ↑
CMR [3]	✓		1	-	-	0.703	81.2
(A)CSM [2]	✓	✓	1	0.622	68.5	0.705	72.4
ACMR [60]	✓		1	-	-	0.708	85.5
TTP *(ours)*	✓		1	**0.656** (0.002)	**70.0** (0.53)	**0.760** (0.004)	**93.4** (0.14)
(A)CSM [2]		✓	8	0.625	**50.9**	0.693	46.8
(A)CSM [2]		✓	1	0.637 (0.004)	39.0 (1.07)	0.684 (0.011)	44.5 (1.21)
TTP *(ours)*			1	**0.652** (0.008)	48.7 (0.66)	**0.752** (0.003)	**50.9** (0.43)

Table 4.2: Performance of TTP method through iterations for pose and deformation estimation. We achieve the best results with more iterations, but even a single iteration suffices for competitive scores.

	Number of Iterations							
	1		2		3		4	
	mIoU	PCK	mIoU	PCK	mIoU	PCK	mIoU	PCK
TTP w/ KP	0.732	92.5	0.755	93.3	0.758	93.3	0.758	93.4
TTP w/o KP	0.746	51.4	0.752	51.1	0.752	51.1	0.752	51.1

a fully connected layer having as input the flattened feature map $\mathbf{z}$ and outputs the regressed 2D positions $\mathbf{u} \in \mathbb{R}^{|V|\times 2}$ and their respective visibility $\mathbf{v} \in \mathbb{R}^{|V|\times 1}$. The number of basis components is set to $K = 16$ and the number of iterations for the camera and deformation estimation is four.

Network Training To train the 3D reconstruction model we first warm up the model without applying any deformation for 100 epochs. This warm-up process allows the model to find the best pose possible given the rigid template using available cues like masks, texture and optional keypoints. We then train the full 3D reconstruction network with deformation enabled and all available cues for another 100 epochs. All experiments were run on a single RTX 2080 Ti GPU.

4.4.2 Quantitative Results

Evaluation on CUB In Table 4.1 we evaluate TTP on the CUB dataset and report the average and standard deviation of 5 experiments with different seeds; the

rigid part of the results table indicates the performance of models that do not use a deformable component, while the non-rigid part amounts to the more challenging problem of estimating both camera and mesh deformations.

We observe that our method outperforms the baseline models on both reported metrics, i.e. mean IoU and keypoint re-projection accuracy, while requiring no camera priors. When using 2D keypoint supervision (upper part of the table) our method achieves the best results, outperforming the closest baseline by almost 8 accuracy points. For the case where no 2D keypoints are used the only published result in the literature is the ACSM approach of [2] which relies on 8 camera hypotheses during both training and testing, while also using manual annotation of part-based rigs to bootstrap the deformation model. Our work outperforms this baseline while relying on a single camera hypothesis and without requiring any manual mesh annotation.

We posit that using multiple cameras in [2] aims at mitigating the local minima in network training and optimization. We have therefore rerun the system of [2] with a single camera and five different optimization seeds and observed a further gap in performance compared to our work, as well as a larger variance in the reconstruction accuracy compared to that of our work for the non-rigid case, suggesting a potentially higher chance of getting stuck in local minima.

Ablation study We ablate various terms in our learning objective and report the mIoU and the semantic keypoint reprojection (PCK) metrics. In particular we examine the impact of removing any of the utilized losses in Table 4.3. When using keypoint supervision the differences in performance are small. However in the absence of keypoints, the method struggles to align the template with the depicted object when we remove the visibility loss. While mIoU remains high, PCK score decreases substantially meaning that the pose and deformation of the template cover the foreground mask when rendered but not from a proper viewpoint of the object. Similar performance drop occurs without the equivariance loss since the method produces pose estimates biased towards one vertical direction. Finally, removing the texture loss causes mIoU performance to drop significantly.

In Table 4.4 we study how performance changes as a function of the number

Table 4.3: Ablation on losses.

	With KP		Without KP	
	mIoU ↑	PCK ↑	mIoU ↑	PCK ↑
TTP	0.765	93.6	0.749	50.9
TTP - L_{pixel}	0.752	92.7	0.667	49.2
TTP - L_{vis}	0.75	92.3	0.74	9.3
TTP - L_{equiv}	0.751	92.5	0.71	28.2

of basis elements. Increasing the number of components tends to increase performance but up to 16 elements for both set of experiments. When training with semantic keypoints increasing the number of basis components further improves performance, however since the same does not apply to the mask-only case we have set $K = 16$ for all of our experiments.

Finally, a key aspect of TTP is the iterative pose and deformation estimation process. In Table 4.2, we provide the mIoU and PCK scores for every iteration for two experiments trained with and without keypoint supervision. Multiple iterations have to be executed to get the best performance, however TTP's performance with a single iteration still outperforms prior work for both metrics.

We are complementing these quantitative results with qualitative results in Figure 4.2 where we show that we can correctly deform the template mesh to produce highly accurate 3D reconstructions. In Figure 4.6 we show the importance of keypoints in supervision. While TTP works with and without keypoints, subtle surface details can only be captured accurately when we use keypoint supervision. For example, the TTP-nokp model struggles to reconstruct long beaks and legs which is attributed to the fact that these areas constitute a small area of the total image. As such, all the 2D losses used for self-supervision don't penalize as much the lack of a long beak in the 3D reconstruction.

4.4.3 Computational Analysis

In Table 4.6 we provide a run time analysis of our and prior methods on the task of self-supervised 3D reconstruction. The analysis was performed on a machine with a NVidia RTX 2080Ti, an Intel Xeon W-2255 and 32 GBs of RAM and reported is

Table 4.4: Ablation on number of basis components.

	With KP		Without KP	
Basis	mIoU ↑	PCK ↑	mIoU ↑	PCK ↑
rigid	0.657	70.7	0.646	48.4
4	0.726	88.2	0.72	47.9
8	0.745	90.6	0.733	50.4
16	0.765	93.6	0.749	50.9
32	0.771	93.7	0.748	49.8
64	0.775	94.2	0.752	49.8

Table 4.5: PASCAL3D+ evaluation. We provide numerical score of TTP with and without keypoint supervision during training. We observe that *even without keypoint supervision* TTP is competitive with the other methods which, except for UCMR, require keypoints.

	DRC [221]	UCMR [59]	CMR [3]	TTP w/ KP	TTP w/o KP
aeroplane	0.42	-	0.468	**0.488**	0.45
car	0.67	0.646	0.64	**0.67**	0.665

the average of 20 runs for the reconstruction of a 256x256 image.

To ensure that the benchmark is fair for all methods, we compute the execution time between the moment an image is given as input to a network up to the moment where the network predicts the mesh, the camera pose and the texture; the implementations of the methods used for comparison are the publicly available ones from the original authors. This ensures that the run times are fairly comparable and they don't reflect additional overhead such as rendering operations or loss computation.

As reported in Table 4.6, TTP is faster than both CSM and ACSM by a considerable margin even on the CPU. CMR is the fastest method of all due to its simplicity, however, it requires keypoint supervision and has substantially lower results as indicated in Table 4.1.

We note that our implementation relies on PyTorch and we have used PyTorch's python-based LBFGS optimizer for convenience and autograd for Jacobian computation; understandably for AR applications the LBFGS timing can become substantially faster in C and with explicitly coded Jacobian matrix computation. This is reflected in Table 4.6 given the fact that the optimization step of TTP re-

Table 4.6: Run time analysis in milliseconds of various self-supervised 3D methods. All benchmarks were run 20 times using images of size 256x256 and we report the average run times.

	Device	Iters	CNN (msec)	Optimization (msec)	Total (msec)
CMR	GPU	-	5.11	N/A	5.11
CSM	GPU	-	150.73	N/A	150.73
ACSM	GPU	-	191.46	N/A	191.46
TTP	GPU	1	4.02	33.44	37.46
		4	4.02	125.14	129.16
TTP	CPU	1	18.52	33.44	51.96
		4	18.52	125.14	143.67

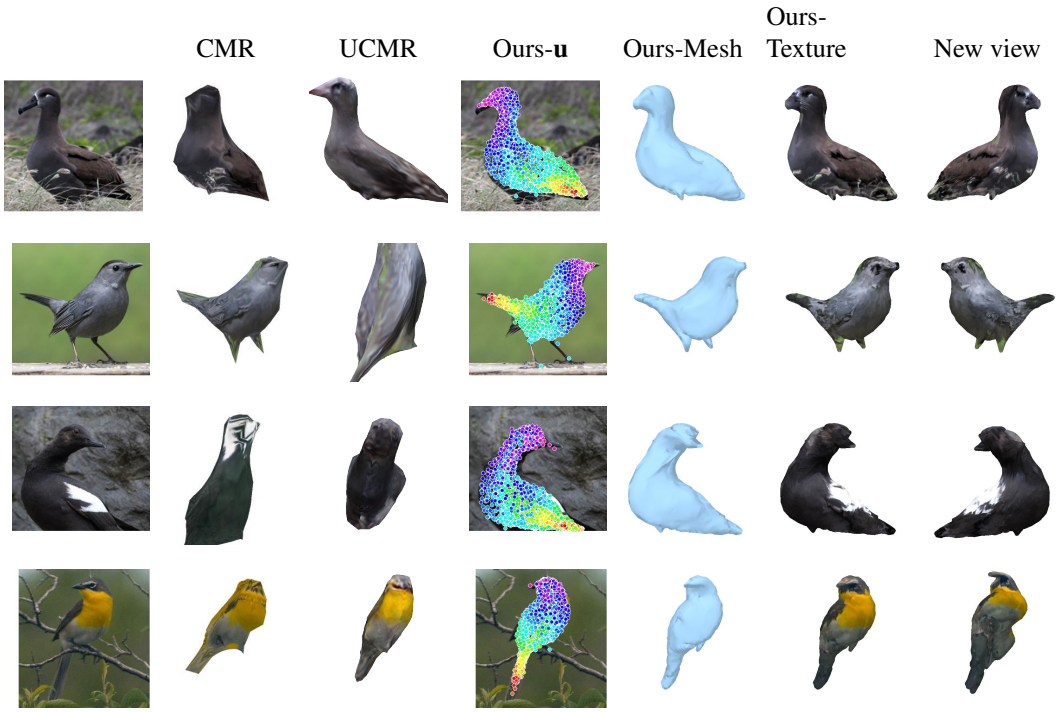


Figure 4.2: Bird reconstructions For each input image we provide the results of CMR [3] and UCMR [2] alongside with our method. We visualize the input image, predictions from prior works and TTP's predicted correspondence (**u**), mesh reconstruction and textured mesh from two viewpoints. We observe that we better capture texture details, deformation and pose estimation.

quires virtually the same time for both CPU and GPU execution. The reason is that PyTorch's LBFGS optimizer doesn't have a CUDA optimized backend and it is not cachable and memory-friendly. This is in return causes the execution speed to be as slow as the CPU execution. The main speed up when changing devices is reflected

only on the neural network regressor due to the optimized CUDA implementations of several operations. Lastly, each optimization step requires approximately 30 seconds so the total execution time can be increased significantly when a lot of steps are used. However, in Table 4.2 we have showed that the accuracy of TTP is competitive even with one iteration and two iterations appear to be a good compromise for both accuracy and speed concerns.

Evaluation on Pascal3D+ While our primary evaluation is on the CUB dataset, we run supplementary experiments on the cars, airplanes and motorcycle categories of PASCAL3D+ dataset. For cars and aeroplanes we provide comparisons against CMR [3], UCMR [59], a volumetric prediction network [221] and a fitting based method [54]. Three of the methods use segmentation masks, cameras and keypoints for supervision except UCMR that does not require keypoints.

We train our method with and without keypoint supervision and provide our 3D mIoU results in Table 4.5. We observe that our method performs considerably better than competing methods even when keypoint supervision is not utilized. Beyond the 3D mIoU metric we also provide the PCK scores of our method for the cars dataset and compare it against the only available reported method [4]. Our approach trained with and without keypoints results in 74.9 mIoU and 45.7 PCK scores while CSM achieves 51.2 and 40.0 respectively. The difference is significant in both cases, especially in the keypoint-free methods where predicting a correct camera pose is challenging due to the weak supervision setup. We provide qualitative results of TTP in Figure 4.3.

4.4.4 Failure Cases

We visualize some failure cases of the proposed method in Figure 4.4. Common failure cases are related to the inability to predict a good camera pose and 2D points. We maintain the belief that increasing the number of training data will allow the proposed self-supervised method, TTP, to achieve better UV points regression as well as camera and deformation predictions.

Figure 4.3: Pascal3D+ results We show predictions of our TTP method for test set images. For each input image we visualize the 3D shape from the predicted camera, the textured shape and a new view.

4.5 Discussion

We have proposed a method to reconstruct 3D meshes, poses and textures of generic objects in the wild without any direct supervision. We learn unsupervised correspondences between 2D image locations and 3D template vertices and use them to compute the camera pose and deformation of the object. Even though our CNN

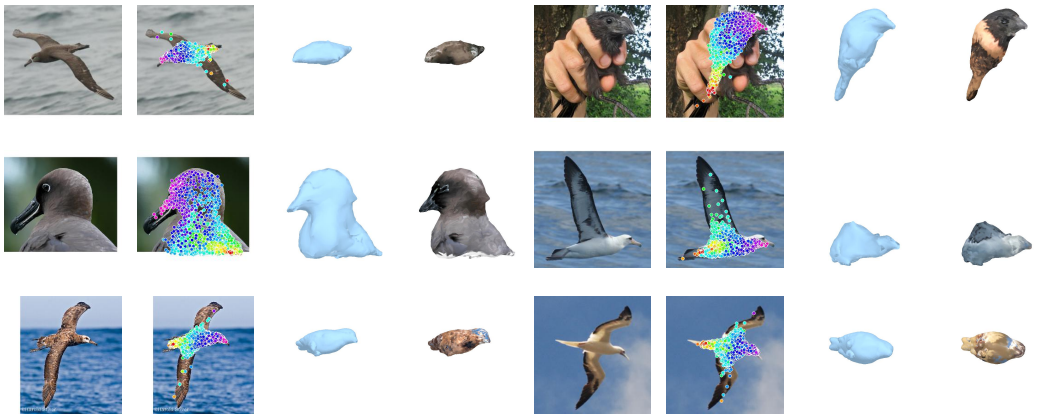


Figure 4.4: Failure Cases: We visualize some failure modes of our method. The columns present the input image, the predicted 2D points, and 3D reconstruction with and without texture.

architecture predicts substantially fewer outputs - compared e.g. to [3] where all of the 3D vertices and the camera are directly regressed by the network, we deliver substantially better results. We attribute this to the use of a direct optimization scheme to optimize the 3D reconstruction problem both during training and testing. The resulting optimization problem is particularly lightweight, meaning that it can be used for interactive applications, e.g. in Augmented Reality, while our results indicate that even a single step of the optimization suffices for accurate mesh recover.

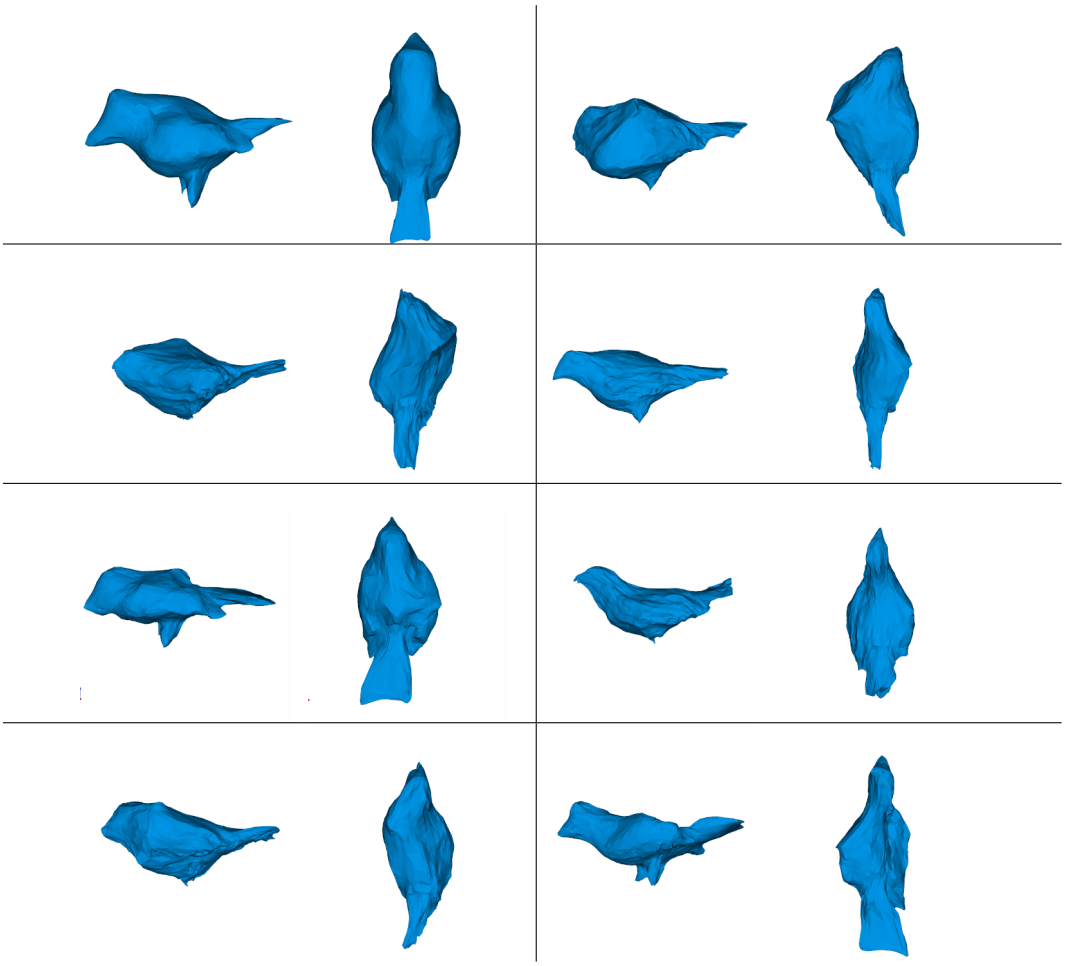


Figure 4.5: Basis Visualization: We visualize 8 basis components $T + B_i$ of a trained TTP experiment for the CUB dataset. For each component we visualize a side and a top view.

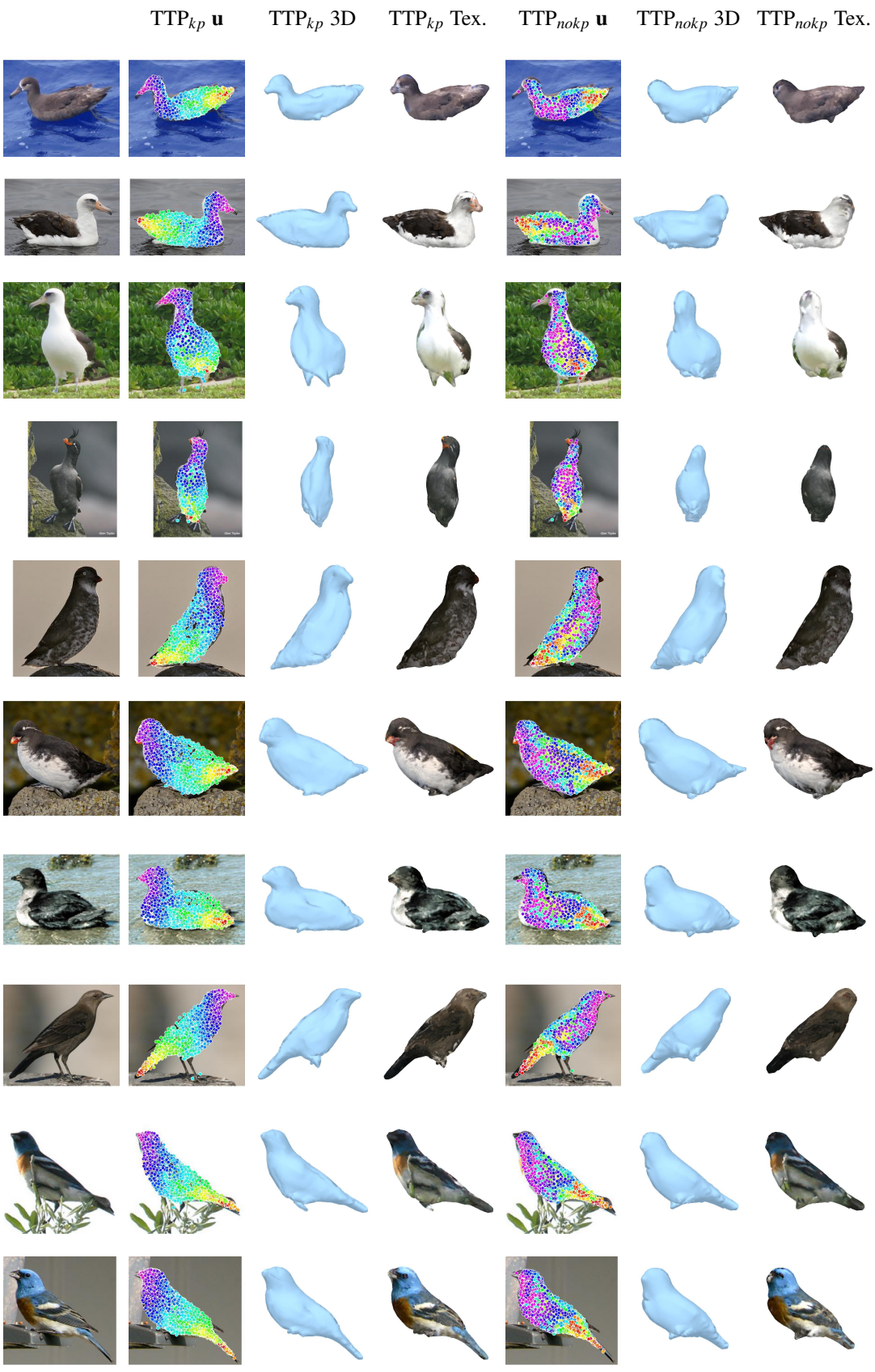


Figure 4.6: Comparison of TTP trained with and without keypoint supervision. We observe that TTP performs equally well on camera pose estimation with and without keypoint supervision. However, keypoint supervision allows for more accurate mesh deformation.

Chapter 5

Concluding Remarks and Future Directions

Differentiable optimization provides the ideal building blocks for an extensive battery of computer vision tasks with countless real-world applications. This thesis explored techniques that enable optimization-based prior knowledge to be used as a core component of computer vision pipelines. In detail, we proposed several methodologies and explored applications related to 2D image reconstruction and 3D monocular reconstruction of arbitrary classes.

The building blocks introduced in each chapter were combined with neural networks to solve 2D and 3D tasks. The proposed methods were quantitatively compared with end-to-end deep learning models, which implicitly solved the examined task. Our findings showcase that optimization-based layers allow networks to accurately solve the problem at hand while also having a transparent algebraic interpretation. For example, in Chapter 2, the transparent interpretation enables us to connect the derived deep learning models directly to the majorization minimization framework and revisit the proposed CNN from an implicit image reconstruction regularization viewpoint. To the same extent, the presented methods in Chapters 3 and 4 enable the development of 3D reconstruction frameworks capable of optimization-based intervention in either deformation, camera pose estimation or both.

Section 5.1 summarizes the contributions of each of the three papers presented

in this thesis and discusses subsequent advances in the field since publication and potential extensions of the work. Moreover, Section 5.2 discusses future directions and critical challenges that were not considered in this thesis and remain currently unsolved to a large extent.

5.1 Summary of contributions

We now highlight the key insights and contributions brought by each method, and finally, we discuss potential extensions of each work.

Chapter 2: Implicit regularization for image reconstruction.

In Chapter 2, we presented an optimization unrolling method with implicit regularization properties for reconstructing images from camera readings. Deviating from deep networks mapping inputs to outputs, we proposed a novel algorithm inspired by powerful classical image regularization methods, large-scale optimization and deep learning techniques. The method is derived from first principles and unrolls a learnable majorization minimization framework fitted to the training data. The network resembles a recurrent neural network and is trained with truncated backpropagation throughout time (TBTT) to circumvent the memory constraints of unrolled methods.

Inspired by the Majorization Minimization framework, the derived neural network has a transparent and clear interpretation as an implicit regularization technique compared to black-box data-driven approaches. The provided experimentation line demonstrates that the network outperforms previous approaches on noisy and noise-free data across different datasets.

The methods that initially outperformed the MMNet architecture used a combination of deeper neural networks [222, 223] and larger datasets [224, 225, 226]. Interestingly in the MSR dataset [119], MMNet maintains at the time of writing the state-of-the-art performance when compared to methods trained on identical data. Beyond image reconstruction from camera readings, similar methods have been developed for image restoration [140, 227, 82, 228, 83, 84], microscopy deblurring [1, 229] and MRI reconstruction [85, 230, 86, 87].

The notion of unrolling optimization schemes has recently been visited from a theoretical standpoint. In [231, 232], the authors examine under which conditions unrolled optimization converges for inverse imaging problems. In detail, the authors in [231] prove the convergence of unrolled schemes under mild assumptions when the deep neural networks are Lipschitz regularized. Similar findings were presented in follow up works with different optimization schemes like fixed-point iteration [233] or proximal algorithms [232, 234].

Chapter 3: Monocular 3D reconstruction with handle-based deformation.

In Chapter 3, we proposed a method that uses self-supervision to 3D reconstruct highly non-rigid objects from images and videos. Monocular 3D reconstruction of articulated object categories is challenging due to the lack of training data and the inherent ill-posedness of the problem. We introduced an interpretable model of 3D template deformations that controls a 3D surface through the displacement of a small number of learnable handles or anchor points. This operation is formulated as a structured layer relying on mesh-laplacian regularization and is trained end-to-end. Interestingly the deformation method boils down to a learnable linear layer where the matrix and the bias term strongly depend on the Laplacian matrix.

The method uses video self-supervision, forcing the consistency of consecutive 3D reconstructions by a motion-based cycle loss. This largely improves both optimization-based and learning-based 3D mesh reconstruction. We also employ a per-sample numerical optimization approach that jointly optimizes over mesh displacements and cameras within a video, boosting accuracy for training and test time post-processing. As shown in Chapter 3, we obtain state-of-the-art reconstructions with diverse shapes, viewpoints and textures for multiple articulated object categories.

One of the weaknesses of the proposed method was the elaborate framework for learning the deformation and camera pose predictor. The framework is based on multiple hypotheses, which increase the computational and memory requirements linearly to the number of hypotheses. This drawback is met in many recent works [4, 2, 59] and Chapter 4 is the first, to our knowledge, approach trying to circumvent it.

Chapter 4: 3D reconstruction with optimization-based camera and deformation estimation.

In Chapter 4, we presented a method that uses self-supervision to 3D reconstruct highly non-rigid objects from images without the need for multiple hypotheses and manual intervention. To The Point (TTP) recovers a 3D shape from a 2D image by first regressing the 2D positions corresponding to the 3D template vertices and then jointly estimating a rigid camera transform and non-rigid template deformation that optimally explain the 2D positions through the 3D shape projection.

The work addresses the multiple hypotheses requirement of the self-supervised method for monocular 3D reconstruction presented in Chapter 3 and similar literature [4, 2, 59]. By relying on 2D-3D correspondences, we use a per-sample optimization problem to replace CNN-based regression of camera pose and non-rigid deformation and obtain substantially more accurate 3D reconstructions. We treat this optimization as a differentiable layer and train the whole system end-to-end while backpropagation through the optimization scheme is achieved via implicit differentiation. We reported systematic quantitative improvements on multiple categories and provided qualitative results comprising diverse shape, pose and texture prediction examples.

At the time of writing, the paper remains the first approach to perform self-supervised 3D reconstruction without multiple-hypothesis for pose and deformation estimation. Future work could make use of continuous 2D to 3D mappings like [4, 2, 204] and a point sampling approach to estimate the camera pose and deformation per sample. We expect that a continuous surface mapping will yield smoother correspondences than the discrete one currently being used and reduce the number of losses needed to train the TTP method.

5.2 Future Directions

Using optimization schemes as layers in machine learning pipelines opens several new research directions and opportunities for influential applications. In the context of the work presented in this thesis, we identify key currently unresolved challenges

and interesting theoretical questions.

Provably converging unrolled optimization schemes with learnable components

The method presented in Chapter 2 is an unrolled optimization scheme with data-driven components. A thorough theoretical investigation would allow us to understand under which conditions the proposed framework converges and bounds the convergence rate. The convergence of optimization schemes depends on whether an operator is contractive. Contractive operators have, by definition, a Lipschitz constant less than 1; however, computing the Lipschitz constant for deep non-linear networks is computationally intractable [235]. Several works [235, 236, 237] have been recently proposed to estimate the Lipschitz constant of generic neural networks, however at the time of writing, the bounds are loose and the computational complexity high. At the same time, works on unrolled optimization schemes for image processing [231, 232] follow a different approach where the neural networks are Lipschitz regularized to enforce bounded Lipschitz constants.

Deploying a deep network once for each outer iteration comes with significant computational overhead, and in most cases, the iterations can not be parallelized. As such, bounding the convergence rate of recently proposed methods [87, 231, 234] provides valuable feedback on how many iterations are needed for an accurate result up to a tolerance.

Unrolled image reconstruction from various signal sources

Another direction is the application of the proposed framework to more ill-posed inverse problems, especially those related to medical signal reconstruction such as Magnetic Resonance Imaging (MRI) [85, 230, 86, 87]. Improvements in image reconstructions technique from medical signals have a direct real-world impact on diagnosis and medical staging. Further applications can be found in microscopes widely used in biological and medical research, allowing the study of organic and inorganic substances at a minuscule scale. The observed microscopy images suffer from the inherent distortion introduced by the imperfections of the imaging system as a whole and by the image-recording sensor in particular. There have been some

recent attempts [1] to address these shortcomings with the image reconstruction process based on learnable unrolled optimization frameworks resembling closely trainable Wiener filters [238].

3D reconstruction of arbitrary objects without strong supervision

Beyond the methodologies presented in this thesis, there are a plethora of different research directions on the applications of structured layers on 3D reconstruction. Both the methods presented in Chapters 3, 4 use structured layers as components of bigger neural networks to push the boundaries of weakly-supervised 3D reconstruction. Using optimization for camera pose estimation alongside neural networks will enable the localization of arbitrary items and scenes. In bundle adjusting inspired Neural Radiance Fields [239, 240] representations of 3D objects are optimized while simultaneously resolving large camera pose misalignments. Camera pose is estimated with bi-level optimization using view synthesis as a proxy objective. Improvements in this area open up exciting avenues for visual localization of SLAM-based systems, a fundamental problem in robotics and self-driving cars.

Another promising direction is minimizing the number of viewpoints required for reconstructing objects. Modern view synthesis methods require hundreds of viewpoints to optimize a 3D radiance field with high fidelity [241]. CNNs trained on billions of images [242, 243] can impose strong priors about objects and minimize the number of viewpoints needed for a photorealistic reconstruction. In [244, 245, 246] the authors present that a deep CNN trained on Imagenet and other datasets regularizes the neural radiance field reconstructions. As a result, even a single viewpoint is enough to generate a 3D reconstruction of an object [246].

Structured layers for Computer Graphics and AR/VR

Algorithms for computer graphics rely on optimization schemes to generate and deform meshes. Techniques like the one presented in Chapters 3 enable us to learn strong priors about the deformation of meshes through multiple frames of a video. An interesting question is whether these techniques could allow professionals in the entertainment industry to deform objects with interactive methods that extrapolate motion learned from extensive collections of data [247, 248, 249].

Instead of modelling the motion frame by frame, the method will predict the motion for the entirety of the time frame [250, 251] while also allowing for interventions from a professional modeller. Given the rising popularity of VR and AR, the same methods could be used for creating interactive filters of animals and objects, which could be projected as a hologram on demand [249].

This thesis presents strong evidence for adapting optimization as a component of deep neural networks with applications in image processing, computational photography, and computer graphics. We have shown that many applications benefit significantly from the paradigm and have pushed the boundaries of the state-of-the-art methods of the targeted applications.

Self-supervised 3D reconstruction of arbitrary objects

The field of 3D reconstruction of generic and deformable objects still faces significant challenges. One major challenge is inferring details from 2D images, as the process is difficult and often results in a lack of precision. This is because, inferring 3D information from 2D images is an ill-posed problem as the same 2D image can be generated from multiple 3D configurations. Additionally, templates and morphable models, which have been used in the past, are not effective for generic classes of objects. These models are limited to specific classes of objects and are not able to generalize to new, unseen classes.

Unsupervised methods such as those proposed in this thesis have been proposed to overcome these limitations, but they still have a long way to go. These methods are based on self-supervised or unsupervised learning, where the model learns from the data itself without any human supervision. But, these methods are still in the early stages of development and are not able to reconstruct objects with high precision and photo-realistic quality. Overall, it is clear that unsupervised 3D methods have a long way to go before they are able to accurately and efficiently reconstruct a wide range of generic and deformable objects. Despite these challenges, ongoing research in the field holds promise for the development of more advanced and effective 3D reconstruction techniques in the future.

Bibliography

[1] Valeriya Pronina, Filippos Kokkinos, Dmitry V. Dylov, and Stamatios Lefkimmiatis. Microscopy image restoration with deep wiener-kolmogorov filters. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 185–201, Cham, 2020. Springer International Publishing.

[2] Nilesh Kulkarni, Abhinav Gupta, David F Fouhey, and Shubham Tulsiani. Articulation-aware canonical surface mapping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 452–461, 2020.

[3] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *ECCV*, 2018.

[4] Nilesh Kulkarni, Abhinav Gupta, and Shubham Tulsiani. Canonical surface mapping via geometric cycle consistency. *International Conference on Computer Vision (ICCV)*, 2019.

[5] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[6] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 1999.

[7] Sami Romdhani and Thomas Vetter. Estimating 3d shape and texture using pixel intensity, edges, specular highlights, texture constraints and a prior. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 986–993. IEEE, 2005.

[8] Xiangyu Zhu, Zhen Lei, Xiaoming Liu, Hailin Shi, and Stan Z Li. Face alignment across large poses: A 3d solution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 146–155, 2016.

[9] Luo Jiang, Juyong Zhang, Bailin Deng, Hao Li, and Ligang Liu. 3d face reconstruction with geometry details from a single image. *IEEE Transactions on Image Processing*, 27(10):4756–4770, 2018.

[10] James Booth, Epameinondas Antonakos, Stylianos Ploumpis, George Trigeorgis, Yannis Panagakis, and Stefanos Zafeiriou. 3d face morphable models" in-the-wild". In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 48–57, 2017.

[11] Tal Hassner and Ronen Basri. Example based 3d reconstruction from single 2d images. In *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*, pages 15–15. IEEE, 2006.

[12] Xudong Cao, Yichen Wei, Fang Wen, and Jian Sun. Face alignment by explicit shape regression. *International journal of computer vision*, 107(2):177–190, 2014.

[13] Tal Hassner. Viewing real-world faces in 3d. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3607–3614, 2013.

[14] Ira Kemelmacher-Shlizerman and Steven M Seitz. Face reconstruction in the wild. In *2011 international conference on computer vision*, pages 1746–1753. IEEE, 2011.

[15] Ira Kemelmacher-Shlizerman and Ronen Basri. 3d face reconstruction from a single image using a single reference face shape. *IEEE transactions on pattern analysis and machine intelligence*, 33(2):394–405, 2010.

[16] Anh Tuan Tran, Tal Hassner, Iacopo Masi, and Gérard Medioni. Regressing robust and discriminative 3d morphable models with a very deep neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5163–5172, 2017.

[17] Elad Richardson, Matan Sela, and Ron Kimmel. 3d face reconstruction by learning from synthetic data. In *2016 fourth international conference on 3D vision (3DV)*, pages 460–469. IEEE, 2016.

[18] Yudong Guo, Jianfei Cai, Boyi Jiang, Jianmin Zheng, et al. Cnn-based real-time dense face reconstruction with inverse-rendered photo-realistic face images. *IEEE transactions on pattern analysis and machine intelligence*, 41(6):1294–1307, 2018.

[19] Stephen Lombardi, Jason Saragih, Tomas Simon, and Yaser Sheikh. Deep appearance models for face rendering. *ACM Transactions on Graphics (ToG)*, 37(4):1–13, 2018.

[20] Baris Gecer, Stylianos Ploumpis, Irene Kotsia, and Stefanos Zafeiriou. Ganfit: Generative adversarial network fitting for high fidelity 3d face reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1155–1164, 2019.

[21] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[22] Cristian Sminchisescu and Alexandru C Telea. Human pose estimation from silhouettes. a consistent approach using distance level sets. In *10th International Conference on Computer Graphics, Visualization and Computer Vision (WSCG'02)*, volume 10, 2002.

[23] Daniel Grest, Dennis Herzog, and Reinhard Koch. Human model fitting from monocular posture images. In *Proc. of VMV*, pages 665–1344. Citeseer, 2005.

[24] Nils Hasler, Hanno Ackermann, Bodo Rosenhahn, Thorsten Thormählen, and Hans-Peter Seidel. Multilinear pose and body shape estimation of dressed subjects from image sets. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1823–1830. IEEE, 2010.

[25] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. In *ACM SIGGRAPH 2005 Papers*, pages 408–416. 2005.

[26] Tejas D Kulkarni, Pushmeet Kohli, Joshua B Tenenbaum, and Vikash Mansinghka. Picture: A probabilistic programming language for scene perception. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 4390–4399, 2015.

[27] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: a skinned multi-person linear model. *ACM Trans. Graph.*, 34(6):248:1–248:16, 2015.

[28] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image. In *Computer Vision – ECCV 2016*, Lecture Notes in Computer Science. Springer International Publishing, October 2016.

[29] Donglai Xiang, Hanbyul Joo, and Yaser Sheikh. Monocular total capture: Posing face, body, and hands in the wild. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10965–10974, 2019.

[30] Georgios Pavlakos, Luyang Zhu, Xiaowei Zhou, and Kostas Daniilidis. Learning to estimate 3d human pose and shape from a single color image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 459–468, 2018.

[31] Angjoo Kanazawa, Jason Y Zhang, Panna Felsen, and Jitendra Malik. Learning 3d human dynamics from video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5614–5623, 2019.

[32] Nikos Kolotouros, Georgios Pavlakos, and Kostas Daniilidis. Convolutional mesh regression for single-image human shape reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4501–4510, 2019.

[33] Nikos Kolotouros, Georgios Pavlakos, Michael J. Black, and Kostas Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[34] Mohamed Omran, Christoph Lassner, Gerard Pons-Moll, Peter Gehler, and Bernt Schiele. Neural body fitting: Unifying deep learning and model based human pose and shape estimation. In *2018 international conference on 3D vision (3DV)*, pages 484–494. IEEE, 2018.

[35] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of Human Shape and Pose. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[36] Silvia Zuffi, Angjoo Kanazawa, David Jacobs, and Michael J. Black. 3D menagerie: Modeling the 3D shape and pose of animals. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[37] Silvia Zuffi, Angjoo Kanazawa, Tanya Y. Berger-Wolf, and Michael J. Black. Three-d safari: Learning to estimate zebra pose, shape, and texture from im-

ages "in the wild". In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, 2019.

[38] Benjamin Biggs, Oliver Boyne, James Charles, Andrew Fitzgibbon, and Roberto Cipolla. Who left the dogs out? 3d animal reconstruction with expectation maximization in the loop. In *European Conference on Computer Vision*, pages 195–211. Springer, 2020.

[39] Berthold KP Horn and Michael J Brooks. The variational approach to shape from shading. *Computer Vision, Graphics, and Image Processing*, 33(2):174–208, 1986.

[40] Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape-from-shading: a survey. *IEEE transactions on pattern analysis and machine intelligence*, 21(8):690–706, 1999.

[41] John Aloimonos. Shape from texture. *Biological cybernetics*, 58(5):345–360, 1988.

[42] Paolo Favaro and Stefano Soatto. A geometric approach to shape from defocus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):406–417, 2005.

[43] Paolo Favaro, Stefano Soatto, Martin Burger, and Stanley J Osher. Shape from defocus via diffusion. *IEEE transactions on pattern analysis and machine intelligence*, 30(3):518–531, 2008.

[44] Andrew Blake and Heinrich Bulthoff. Shape from specularities: Computation and psychophysics. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 331(1260):237–252, 1991.

[45] Jonathan T Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. *IEEE transactions on pattern analysis and machine intelligence*, 37(8):1670–1687, 2014.

[46] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 205–214, 1987.

[47] Demetri Terzopoulos, Andrew Witkin, and Michael Kass. Symmetry-seeking models and 3d object reconstruction. *International Journal of Computer Vision*, 1(3):211–221, 1988.

[48] Mukta Prasad and Andrew Fitzgibbon. Single view reconstruction of curved surfaces. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pages 1345–1354. IEEE, 2006.

[49] Martin R Oswald, Eno Töppe, and Daniel Cremers. Fast and globally optimal single view reconstruction of curved objects. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 534–541. IEEE, 2012.

[50] Sara Vicente and Lourdes Agapito. Balloon shapes: Reconstructing and deforming objects with volume from images. In *2013 International Conference on 3D Vision-3DV 2013*, pages 223–230. IEEE, 2013.

[51] Mathieu Perriollat, Richard Hartley, and Adrien Bartoli. Monocular template-based reconstruction of inextensible surfaces. *International journal of computer vision*, 95(2):124–137, 2011.

[52] Nail Gumerov, Ali Zandifar, Ramani Duraiswami, and Larry S Davis. Structure of applicable surfaces from single views. In *European conference on computer vision*, pages 482–496. Springer, 2004.

[53] Joao Carreira, Abhishek Kar, Shubham Tulsiani, and Jitendra Malik. Virtual view networks for object reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2937–2946, 2015.

[54] Abhishek Kar, Shubham Tulsiani, Joao Carreira, and Jitendra Malik. Category-specific object reconstruction from a single image. In *Proceedings*

of the IEEE conference on computer vision and pattern recognition, pages 1966–1974, 2015.

[55] F. Kokkinos and S. Lefkimmiatis. Iterative joint image demosaicking and denoising using a residual denoising network. *IEEE Transactions on Image Processing*, 28(8):4177–4188, 2019.

[56] Sara Vicente, João Carreira, de Lourdes Agapito, and Jorge Batista. Reconstructing pascal voc. *Computer Vision and Pattern Recognition*, pages 41–48, 2014.

[57] Thomas J Cashman and Andrew W Fitzgibbon. What shape are dolphins? building 3d morphable models from 2d images. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):232–244, 2012.

[58] Bernhard Reinert, Tobias Ritschel, and Hans-Peter Seidel. Animated 3d creatures from single-view video by skeletal sketching. In *Graphics Interface*, pages 133–141, 2016.

[59] Shubham Goel, Angjoo Kanazawa, , and Jitendra Malik. Shape and viewpoints without keypoints. In *ECCV*, 2020.

[60] Xueting Li, Sifei Liu, Shalini De Mello, Kihwan Kim, Xiaolong Wang, Ming-Hsuan Yang, and Jan Kautz. Online adaptation for consistent mesh reconstruction in the wild. In *NeurIPS*, 2020.

[61] Filippos Kokkinos and Iason Kokkinos. Learning monocular 3d reconstruction of articulated categories from motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[62] Mihir Sahasrabudhe, Zhixin Shu, Edward Bartrum, Riza Alp Guler, Dimitris Samaras, and Iasonas Kokkinos. Lifting autoencoders: Unsupervised learning of a fully-disentangled 3d morphable model using deep non-rigid structure from motion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, Oct 2019.

[63] Xueting Li, Sifei Liu, Kihwan Kim, Shalini De Mello, Varun Jampani, Ming-Hsuan Yang, and Jan Kautz. Self-supervised Single-view 3D Reconstruction via Semantic Consistency. In *ECCV*, 2020.

[64] Robert W Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Transactions on graphics (TOG)*, 23(3):399–405, 2004.

[65] Robert W Sumner, Matthias Zwicker, Craig Gotsman, and Jovan Popović. Mesh-based inverse kinematics. *ACM transactions on graphics (TOG)*, 24(3):488–495, 2005.

[66] Scott Schaefer and Can Yuksel. Example-based skeleton extraction. In *Symposium on Geometry Processing*, pages 153–162, 2007.

[67] Stephen W Bailey, Dave Otte, Paul Dilorenzo, and James F O'Brien. Fast and deep deformation approximations. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, 2018.

[68] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116, 2007.

[69] Yaron Lipman, Olga Sorkine, Daniel Cohen-Or, David Levin, Christian Rossi, and Hans-Peter Seidel. Differential coordinates for interactive mesh editing. In *Proceedings Shape Modeling Applications, 2004.*, pages 181–190. IEEE, 2004.

[70] Stéphane Calderon and Tamy Boubekeur. Bounding proxies for shape approximation. *ACM Trans. Graph.*, 36(4):57–1, 2017.

[71] Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. Harmonic coordinates for character articulation. *ACM Transactions on Graphics (TOG)*, 26(3):71–es, 2007.

[72] Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangular meshes. In *ACM Siggraph 2005 Papers*, pages 561–566. 2005.

[73] Yaron Lipman, David Levin, and Daniel Cohen-Or. Green coordinates. *ACM Transactions on Graphics (TOG)*, 27(3):1–10, 2008.

[74] Wang Yifan, Noam Aigerman, Vladimir G Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural cages for detail-preserving 3d deformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 75–83, 2020.

[75] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[76] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[77] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

[78] Justin Domke. Generic methods for optimization-based modeling. In *Artificial Intelligence and Statistics*, pages 318–326, 2012.

[79] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45, 1989.

[80] Liang-Chieh Chen, Alexander Schwing, Alan Yuille, and Raquel Urtasun. Learning deep structured models. In *International Conference on Machine Learning*, pages 1785–1794, 2015.

[81] Uwe Schmidt and Stefan Roth. Shrinkage fields for effective image restoration. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2774–2781, 2014.

[82] Raied Aljadaany, Dipan K Pal, and Marios Savvides. Douglas-rachford networks: Learning both the image prior and data fidelity terms for blind image deconvolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10235–10244, 2019.

[83] Bruno Lecouat, Jean Ponce, and Julien Mairal. Fully trainable and interpretable non-local sparse models for image restoration. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*, pages 238–254. Springer, 2020.

[84] Zhonghao Zhang, Yipeng Liu, Jiani Liu, Fei Wen, and Ce Zhu. Amp-net: Denoising-based deep unfolding for compressive image sensing. *IEEE Transactions on Image Processing*, 30:1487–1500, 2020.

[85] Jian Zhang and Bernard Ghanem. Ista-net: Iterative Shrinkage-Thresholding Algorithm Inspired Deep Network for Image Compressive Sensing. *CoRR*, abs/1706.07929, 2017.

[86] Yiling Liu, Qiegen Liu, Minghui Zhang, Qingxin Yang, Shanshan Wang, and Dong Liang. Ifr-net: Iterative feature refinement network for compressed sensing mri. *IEEE Transactions on Computational Imaging*, 6:434–446, 2019.

[87] Tieyuan Lu, Xinlin Zhang, Yihui Huang, Di Guo, Feng Huang, Qin Xu, Yuhan Hu, Lin Ou-Yang, Jianzhong Lin, Zhiping Yan, et al. pfista-sense-resnet for parallel mri reconstruction. *Journal of Magnetic Resonance*, 318:106790, 2020.

[88] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Con-

ditional random fields as recurrent neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1529–1537, 2015.

[89] Alexander G Schwing and Raquel Urtasun. Fully connected deep structured networks. *arXiv preprint arXiv:1503.02351*, 2015.

[90] Ziwei Liu, Xiaoxiao Li, Ping Luo, Chen-Change Loy, and Xiaoou Tang. Semantic image segmentation via deep parsing network. In *Proceedings of the IEEE international conference on computer vision*, pages 1377–1385, 2015.

[91] Anurag Arnab, Sadeep Jayasumana, Shuai Zheng, and Philip HS Torr. Higher order conditional random fields in deep neural networks. In *European Conference on Computer Vision*, pages 524–540. Springer, 2016.

[92] Siddhartha Chandra and Iasonas Kokkinos. Fast, exact and multi-scale inference for semantic image segmentation with deep gaussian crfs. 2016.

[93] Siddhartha Chandra, Nicolas Usunier, and Iasonas Kokkinos. Dense and low-rank gaussian crfs using deep embeddings. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[94] Jatavallabhula Krishna Murthy, Soroush Saryazdi, Ganesh Iyer, and Liam Paull. gradslam: Dense slam meets automatic differentiation. In *arXiv*, 2020.

[95] Asen L Dontchev and R Tyrrell Rockafellar. *Implicit functions and solution mappings*, volume 543. Springer, 2009.

[96] Julien Mairal, Francis Bach, and Jean Ponce. Task-driven dictionary learning. *IEEE transactions on pattern analysis and machine intelligence*, 34(4):791–804, 2011.

[97] Stephen Gould, Richard Hartley, and Dylan Campbell. Deep declarative networks, 2021.

[98] Dylan Campbell*, Liu Liu*, and Stephen Gould. Solving the blind perspective-n-point problem end-to-end with robust differentiable geometric optimization. In *ECCV*, 2020. * equal contribution.

[99] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[100] Basura Fernando and Stephen Gould. Learning end-to-end video classification with rank-pooling. In *International Conference on Machine Learning*, pages 1187–1196. PMLR, 2016.

[101] Basura Fernando and Stephen Gould. Discriminatively learned hierarchical rank pooling networks. *International Journal of Computer Vision*, 124(3):335–355, 2017.

[102] Basura Fernando, Efstratios Gavves, Jose M Oramas, Amir Ghodrati, and Tinne Tuytelaars. Modeling video evolution for action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5378–5387, 2015.

[103] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10657–10665, 2019.

[104] Filippos Kokkinos and Stamatios Lefkimmiatis. Deep image demosaicking using a cascade of convolutional residual denoising networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[105] Filippos Kokkinos and Stamatis Lefkimmiatis. Iterative residual cnns for burst photography applications. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[106] Filippos Kokkinos and Iasonas Kokkinos. To the point: Correspondence-driven monocular 3d category reconstruction. In *Advances in Neural Information Processing Systems*, 2021.

[107] Francesca Babiloni, Ioannis Marras, Filippos Kokkinos, Jiankang Deng, Grigorios Chrysos, and Stefanos Zafeiriou. Poly-nl: Linear complexity non-local layers with polynomials. *arXiv preprint arXiv:2107.02859*, 2021.

[108] Filippos Kokkinos, Ioannis Marras, Matteo Maggioni, Gregory G. Slabaugh, and Stefanos Zafeiriou. Pixel adaptive filtering units. *CoRR*, abs/1911.10581, 2019.

[109] Lei Zhang Xin Li, Bahadir Gunturk. Image demosaicing: a systematic survey. *Proc.SPIE*, 6822:6822–6822–15, 2008.

[110] Lei Zhang, Xiaolin Wu, Antoni Buades, and Xin Li. Color demosaicking by local directional interpolation and nonlocal adaptive thresholding. *Journal of Electronic imaging*, 20(2):023016, 2011.

[111] Joan Duran and Antoni Buades. Self-similarity and spectral correlation adaptive algorithm for color demosaicking. *IEEE transactions on image processing*, 23(9):4031–4040, 2014.

[112] Antoni Buades, Bartomeu Coll, Jean-Michel Morel, and Catalina Sbert. Self-similarity driven color demosaicking. *IEEE Transactions on Image Processing*, 18(6):1192–1202, 2009.

[113] Felix Heide, Markus Steinberger, Yun-Ta Tsai, Mushfiqur Rouf, Dawid Pajak, Dikpal Reddy, Orazio Gallo, Jing Liu, Wolfgang Heidrich, Karen Egiazarian, et al. Flexisp: A flexible camera image processing framework. *ACM Transactions on Graphics (TOG)*, 33(6):231, 2014.

[114] Kan Chang, Pak Lun Kevin Ding, and Baoxin Li. Color image demosaicking using inter-channel correlation and nonlocal self-similarity. *Signal Processing: Image Communication*, 39:264–279, 2015.

[115] D. S. Tan, W. Chen, and K. Hua. Deepdemosaicking: Adaptive image demosaicking via multiple deep fully convolutional networks. *IEEE Transactions on Image Processing*, 27(5):2408–2419, May 2018.

[116] D. Alleysson, S. Susstrunk, and J. Herault. Linear demosaicing inspired by the human visual system. *IEEE Transactions on Image Processing*, 14(4):439–449, April 2005.

[117] J. Sun and M. F. Tappen. Separable Markov Random Field Model and Its Applications in low level vision. *IEEE Transactions on Image Processing*, 22(1):402–407, Jan 2013.

[118] Fang-Lin He, Yu-Chiang Frank Wang, and Kai-Lung Hua. Self-learning approach to color demosaicking via support vector regression. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 2765–2768. IEEE, 2012.

[119] D. Khashabi, S. Nowozin, J. Jancsary, and A. W. Fitzgibbon. Joint Demosaicing and Denoising via Learned Nonparametric random fields. *IEEE Transactions on Image Processing*, 23(12):4968–4981, Dec 2014.

[120] Hagit Zabrodsky Hel-Or Oren Kapah. Demosaicking using artificial neural networks. *Proc.SPIE*, 3962:3962 – 3962 – 9, 2000.

[121] Jinwook Go and Chulhee Lee. Interpolation using neural network for digital still cameras. *IEEE*, pages 176–177, June 2000.

[122] Y. Wang. A multilayer neural network for image demosaicking. *IEEE*, pages 1852–1856, Oct 2014.

[123] Michaël Gharbi, Gaurav Chaurasia, Sylvain Paris, and Frédo Durand. Deep Joint Demosaicking and Denoising. *ACM Trans. Graph.*, 35(6):191:1–191:12, November 2016.

[124] Bernardo Henz, Eduardo S. L. Gastal, and Manuel M. Oliveira. Deep joint design of color filter arrays and demosaicing. *Computer Graphics Forum*, 37(2):389–399, 2018.

[125] A. Foi, M. Trimeche, V. Katkovnik, and K. Egiazarian. Practical Poissonian-gaussian noise modeling and fitting for single-image raw-data. *IEEE Transactions on Image Processing*, 17(10):1737–1754, Oct 2008.

[126] Henry Rantanen Ossi Kalevo. Noise reduction techniques for bayer-matrix images. *Proc.SPIE*, 4669:4669 – 4669 – 12, 2002.

[127] S. Farsiu, M. Elad, and P. Milanfar. Multiframe demosaicing and super-resolution of color images. *IEEE Transactions on Image Processing*, 15(1):141–159, Jan 2006.

[128] D. Menon and G. Calvagno. Joint demosaicking and denoisingwith space-varying filters. In *2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 477–480, Nov 2009.

[129] L. Zhang, R. Lukac, X. Wu, and D. Zhang. Pca-based spatially adaptive denoising of CFA images for Single-Sensor Digital Cameras. *IEEE Transactions on Image Processing*, 18(4):797–812, April 2009.

[130] T. Klatzer, K. Hammernik, P. Knobelreiter, and T. Pock. Learning joint demosaicing and denoising based on sequential energy minimization. In *2016 IEEE International Conference on Computational Photography (ICCP)*, pages 1–11, May 2016.

[131] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[132] Tom Goldstein and Stanley Osher. The split Bregman method for l1-regularized problems. *SIAM journal on imaging sciences*, 2(2):323–343, 2009.

[133] David R Hunter and Kenneth Lange. A tutorial on MM algorithms. *The American Statistician*, 58(1):30–37, 2004.

[134] Mário AT Figueiredo, José M Bioucas-Dias, and Robert D Nowak. Majorization–minimization algorithms for wavelet-based image restoration. *IEEE Transactions on Image processing*, 16(12):2980–2991, 2007.

[135] S. Lefkimmiatis, P. Ward, and M. Unser. Hessian Schatten-norm regularization for linear inverse problems. *IEEE Transactions on Image processing*, 22(5):1873–1888, 2013.

[136] Yaniv Romano, Michael Elad, and Peyman Milanfar. The little engine that could: Regularization by denoising (RED). *SIAM Journal on Imaging Sciences*, 10(4):1804–1844, 2017.

[137] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg. Plug-and-Play priors for model based reconstruction. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 945–948, Dec 2013.

[138] K. Zhang, W. Zuo, S. Gu, and L. Zhang. Learning Deep CNN Denoiser Prior for Image Restoration. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2808–2817, July 2017.

[139] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, July 2017.

[140] Stamatios Lefkimmiatis. Universal denoising networks : A Novel CNN Architecture for Image Denoising. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[141] Alessandro Foi. Clipped noisy images: Heteroskedastic modeling and practical denoising. *Signal Processing*, 89(12):2609–2629, 2009.

[142] Xinhao Liu, Masayuki Tanaka, and Masatoshi Okutomi. Single-image noise level estimation for blind denoising. *IEEE transactions on image processing*, 22(12):5226–5237, 2013.

[143] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[144] Amir Beck and Marc Teboulle. A fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[145] Huan Li and Zhouchen Lin. Accelerated Proximal Gradient Methods for Nonconvex Programming. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 379–387. Curran Associates, Inc., 2015.

[146] Qihang Lin and Lin Xiao. An adaptive accelerated proximal gradient method and its homotopy continuation for sparse optimization. *Computational Optimization and Applications*, 60(3):633–674, Apr 2015.

[147] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 60–65. IEEE, 2005.

[148] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007.

[149] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 416–423 vol.2, 2001.

[150] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.

In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[151] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of Adam and Beyond. In *International Conference on Learning Representations*, 2018.

[152] A. J. Robinson and Frank Fallside. The Utility Driven Dynamic Error Propagation Network. Technical Report CUED/F-INFENG/TR.1, Engineering Department, Cambridge University, Cambridge, UK, 1987.

[153] Weisheng Dong, Peiyao Wang, Wotao Yin, Guangming Shi, Fangfang Wu, and Xiaotong Lu. Denoising Prior Driven Deep Neural Network for Image Restoration. *CoRR*, abs/1801.06756, 2018.

[154] Anat Levin, Boaz Nadler, Fredo Durand, and William T. Freeman. Patch complexity, finite pixel correlations and optimal denoising. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, pages 73–86, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[155] K. Hirakawa and T. W. Parks. Adaptive homogeneity-directed demosaicing algorithm. *IEEE Transactions on Image Processing*, 14(3):360–369, March 2005.

[156] P. Getreuer. Color demosaicing with contour stencils. In *2011 17th International Conference on Digital Signal Processing (DSP)*, pages 1–6, July 2011.

[157] Filippos Kokkinos and Stamatios Lefkimmiatis. Deep Image Demosaicking using a Cascade of Convolutional Residual Denoising Networks. In *ECCV*. Springer, September 2018.

[158] David L Donoho and Jain M Johnstone. Ideal spatial adaptation by wavelet shrinkage. *biometrika*, 81(3):425–455, 1994.

[159] Riza Alp Guler and Iasonas Kokkinos. Holopose: Holistic 3d human reconstruction in-the-wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[160] Donglai Xiang, Hanbyul Joo, and Yaser Sheikh. Monocular total capture: Posing face, body, and hands in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 10965–10974. Computer Vision Foundation / IEEE, 2019.

[161] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[162] Georgios Pavlakos, Nikos Kolotouros, and Kostas Daniilidis. Texturepose: Supervising human mesh estimation with texture consistency. In *International Conference on Computer Vision, ICCV*, 2019.

[163] Hanbyul Joo, Natalia Neverova, and Andrea Vedaldi. Exemplar fine-tuning for 3d human model fitting towards in-the-wild 3d human pose estimation, 2020.

[164] Vasileios Choutas, Georgios Pavlakos, Timo Bolkart, Dimitrios Tzionas, and Michael J. Black. Monocular expressive body regression through body-driven attention. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part X*, 2020.

[165] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *International journal of computer vision*, 9(2):137–154, 1992.

[166] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[167] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, pages 225–234. IEEE, 2007.

[168] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*. IEEE, 2011.

[169] Lorenzo Torresani, Aaron Hertzmann, and Christoph Bregler. Learning non-rigid 3D shape from 2D motion. *Advances in neural information processing systems*, 2003.

[170] Ravi Garg, Anastasios Roussos, and Lourdes Agapito. Dense variational reconstruction of non-rigid surfaces from monocular video. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[171] Katerina Fragkiadaki, Han Hu, and Jianbo Shi. Pose from flow and flow from pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2059–2066, 2013.

[172] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and H-P Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184, 2004.

[173] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle Adjustment - A Modern Synthesis. In *Vision Algorithms: Theory and Practice*, 1999.

[174] Alec Jacobson, Ilya Baran, Jovan Popovic, and Olga Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.*, 2011.

[175] Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. Fast automatic skinning transformations. *ACM Transactions on Graphics (TOG)*, 2012.

[176] P. Tokmakov, K. Alahari, and C. Schmid. Weakly-supervised semantic segmentation using motion cues. In *ECCV*, 2016.

[177] Natalia Neverova, James Thewlis, Riza Alp Güler, Iasonas Kokkinos, and Andrea Vedaldi. Slim densepose: Thrifty learning from sparse annotations and motion cues. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, 2019.

[178] Hsiao-Yu Tung, Hsiao-Wei Tung, Ersin Yumer, and Katerina Fragkiadaki. Self-supervised learning of motion capture. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5236–5246. Curran Associates, Inc., 2017.

[179] Thiemo Alldieck, Marc Kassubeck, Bastian Wandt, Bodo Rosenhahn, and Marcus Magnor. Optical flow-based 3d human motion estimation from monocular video. In Volker Roth and Thomas Vetter, editors, *Pattern Recognition*, pages 347–360, Cham, 2017. Springer International Publishing.

[180] Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, and Katerina Fragkiadaki. Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*, 2017.

[181] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017.

[182] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. Demon: Depth and motion network for learning monocular stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[183] Yana Hasson, Bugra Tekin, Federica Bogo, Ivan Laptev, Marc Pollefeys, and Cordelia Schmid. Leveraging photometric consistency over time for sparsely supervised hand-object reconstruction. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[184] Shengyu Zhao, Yilun Sheng, Yue Dong, Eric I-Chao Chang, and Yan Xu. MaskFlownet: Asymmetric Feature Matching with Learnable Occlusion Mask. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[185] Qi xing Huang and Leonidas Guibas. Consistent shape maps via semidefinite programming. In *In Computer Graphics Forum*, pages 177–186. Wiley Online Library, 2013.

[186] Tinghui Zhou, Philipp Krahenbuhl, Mathieu Aubry, Qixing Huang, and Alexei A Efros. Learning dense correspondence via 3d-guided cycle consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 117–126, 2016.

[187] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[188] Gabriel Taubin. A signal processing approach to fair surface design. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 351–358, 1995.

[189] Paul L Fackler. Notes on matrix calculus. *Privately Published*, 2005.

[190] Y Eldar. *Irregular image sampling using the Voronoi diagram*. PhD thesis, M. Sc. thesis, Technion-IIT, Israel, 1992.

[191] Dominik Kulon, Riza Alp Guler, Iasonas Kokkinos, Michael M. Bronstein, and Stefanos Zafeiriou. Weakly-supervised mesh-convolutional hand reconstruction in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[192] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[193] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020.

[194] L. Del Pero, S. Ricco, R. Sukthankar, and V. Ferrari. Articulated motion discovery using pairs of trajectories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[195] Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5188–5197, 2019.

[196] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[197] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7297–7306, 2018.

[198] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.

[199] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[200] Rui Yu, Chris Russell, Neill D. F. Campbell, and Lourdes Agapito. Direct, dense, and deformable: Template-based non-rigid 3d reconstruction from rgb video. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[201] Ijaz Akhter, Yaser Sheikh, Sohaib Khan, and Takeo Kanade. Nonrigid structure from motion in trajectory space. In D. Koller, D. Schuurmans, Y. Bengio,

and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2009.

[202] Marco Paladini, Alessio Del Bue, Marko Stosic, Marija Dodig, Joao Xavier, and Lourdes Agapito. Factorization for non-rigid and articulated structure using metric projections. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2898–2905. IEEE, 2009.

[203] Yuchao Dai, Hongdong Li, and Mingyi He. A simple prior-free method for non-rigid structure-from-motion factorization. *International Journal of Computer Vision*, 107(2):101–122, 2014.

[204] David Novotny, Roman Shapovalov, and Andrea Vedaldi. Canonical 3D Deformer Maps: Unifying parametric and non-parametric methods for dense weakly-supervised category reconstruction. In *NeurIPS*, 2020.

[205] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and Z. Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, 2019.

[206] Stephen Gould, Richard Hartley, and Dylan Campbell. Deep declarative networks: A new hope. Technical report, Australian National University (arXiv:1909.04866), Sep 2019.

[207] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2014.

[208] Simon Jenni and Paolo Favaro. Self-supervised multi-view synchronization learning for 3d pose estimation. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[209] Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Unsupervised learning of probably symmetric deformable 3D objects from images in the

wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[210] David Novotny, Nikhila Ravi, Benjamin Graham, Natalia Neverova, and Andrea Vedaldi. C3dpo: Canonical 3d pose networks for non-rigid structure from motion. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[211] Chen Kong and Simon Lucey. Deep non-rigid structure from motion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1558–1567, 2019.

[212] Katerina Fragkiadaki, Marta Salas, Pablo Andres Arbelaez, and Jitendra Malik. Grouping-based low-rank trajectory completion and 3d reconstruction. In *NIPS*, volume 2, page 7. Citeseer, 2014.

[213] Xiaowei Zhou, Menglong Zhu, Spyridon Leonardos, and Kostas Daniilidis. Sparse representation for 3d shape estimation: A convex relaxation approach. *IEEE transactions on pattern analysis and machine intelligence*, 39(8):1648–1661, 2016.

[214] Xiaowei Zhou, Menglong Zhu, Spyridon Leonardos, Konstantinos G Derpanis, and Kostas Daniilidis. Sparseness meets deepness: 3d human pose estimation from monocular video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4966–4975, 2016.

[215] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[216] Bo Chen, Alvaro Parra, Jiewei Cao, Nan Li, and Tat-Jun Chin. End-to-end learnable geometric vision by backpropagating pnp optimization. In *CVPR*, 2020.

[217] Marcin Andrychowicz, Misha Denil, Sergio Gómez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[218] Filippos Kokkinos and Stamatios Lefkimmiatis. Iterative joint image demosaicking and denoising using a residual denoising network. *IEEE Transactions on Image Processing*, 28(8):4177–4188, 2019.

[219] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. Dsac-differentiable ransac for camera localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6684–6692, 2017.

[220] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[221] Shubham Tulsiani, Tinghui Zhou, Alexei A. Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Computer Vision and Pattern Regognition (CVPR)*, 2017.

[222] Lin Liu, Xu Jia, Jianzhuang Liu, and Qi Tian. Joint demosaicing and denoising with self guidance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2240–2249, 2020.

[223] Thibaud Ehret, Axel Davy, Pablo Arias, and Gabriele Facciolo. Joint demosaicking and denoising by fine-tuning of bursts of raw images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8868–8877, 2019.

[224] Jie Tang, Jian Li, and Ping Tan. Demosaicing by differentiable deep restoration. *Applied Sciences*, 11(4):1649, 2021.

[225] Takuro Yamaguchi and Masaaki Ikehara. Multi-stage dense cnn demosaicking with downsampling and re-indexing structure. *IEEE Access*, 8:175160–175168, 2020.

[226] Jierun Chen, Song Wen, and S-H Gary Chan. Joint demosaicking and denoising in the wild: The case of training under ground truth uncertainty. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1018–1026, 2021.

[227] Kai Zhang, Yawei Li, Wangmeng Zuo, Lei Zhang, Luc Van Gool, and Radu Timofte. Plug-and-play image restoration with deep denoiser prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[228] Yu Sun, Zihui Wu, Xiaojian Xu, Brendt Wohlberg, and Ulugbek S Kamilov. Scalable plug-and-play admm with convergence guarantees. *IEEE Transactions on Computational Imaging*, 7:849–863, 2021.

[229] Anatasiia Kornilova, Mikhail Salnikov, Olga Novitskaya, Maria Begicheva, Egor Sevriugov, Kirill Shcherbakov, Valeriya Pronina, and Dmitry V Dylov. Deep learning framework for mobile microscopy. In *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, pages 324–328. IEEE, 2021.

[230] Jing Cheng, Haifeng Wang, Leslie Ying, and Dong Liang. Model learning: Primal dual networks for fast mr imaging. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 21–29. Springer, 2019.

[231] Ernest Ryu, Jialin Liu, Sicheng Wang, Xiaohan Chen, Zhangyang Wang, and Wotao Yin. Plug-and-play methods provably converge with properly trained denoisers. In *International Conference on Machine Learning*, pages 5546–5557. PMLR, 2019.

[232] Kaixuan Wei, Angelica Aviles-Rivero, Jingwei Liang, Ying Fu, Carola-Bibiane Schönlieb, and Hua Huang. Tuning-free plug-and-play proximal

algorithm for inverse imaging problems. In *International Conference on Machine Learning*, pages 10158–10169. PMLR, 2020.

[233] Regev Cohen, Michael Elad, and Peyman Milanfar. Regularization by denoising via fixed-point projection (red-pro). *SIAM Journal on Imaging Sciences*, 14(3):1374–1406, 2021.

[234] Jialin Liu and Xiaohan Chen. Alista: Analytic weights are as good as learned weights in lista. In *International Conference on Learning Representations (ICLR)*, 2019.

[235] Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[236] Dongmian Zou, Radu Balan, and Maneesh Singh. On lipschitz bounds of general convolutional neural networks. *IEEE Transactions on Information Theory*, 66(3):1738–1759, 2019.

[237] Hyunjik Kim, George Papamakarios, and Andriy Mnih. The lipschitz constant of self-attention. In *International Conference on Machine Learning*, pages 5562–5571. PMLR, 2021.

[238] Norbert Wiener et al. *Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications*, volume 8. MIT press Cambridge, MA, 1964.

[239] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[240] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. NeRF−−: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021.

[241] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[242] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[243] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR, 18–24 Jul 2021.

[244] Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5885–5894, 2021.

[245] Ajay Jain, Ben Mildenhall, Jonathan T. Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. December 2021.

[246] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021.

[247] Daniel Holden, Taku Komura, and Jun Saito. Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)*, 36(4):1–13, 2017.

[248] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

[249] Ruilong Li, Shan Yang, David A Ross, and Angjoo Kanazawa. Ai choreographer: Music conditioned 3d dance generation with aist++. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13401–13412, 2021.

[250] Gengshan Yang, Deqing Sun, Varun Jampani, Daniel Vlasic, Forrester Cole, Ce Liu, and Deva Ramanan. Viser: Video-specific surface embeddings for articulated 3d shape reconstruction. In *NeurIPS*, 2021.

[251] Gengshan Yang, Deqing Sun, Varun Jampani, Daniel Vlasic, Forrester Cole, Huiwen Chang, Deva Ramanan, William T Freeman, and Ce Liu. Lasr: Learning articulated shape reconstruction from a monocular video. In *CVPR*, 2021.