UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Bridging the gap between reconstruction and synthesis

Albert Pumarola Peris

Universitat Politècnica de Catalunya

Doctoral Programme

Automatic Control, Robotics and Computer Vision

Ph.D. Thesis

Bridging the gap between Reconstruction and Synthesis

Albert Pumarola Peris

Advisors:
Francesc Moreno Noguer and Alberto Sanfeliu Cortés

Barcelona, April 2021

Bridging the gap between Reconstruction and Synthesis

A thesis submitted to the Universitat Politècnica de Catalunya
to obtain the degree of Doctor of Philosophy

Doctoral programme:
Automatic Control, Robotics and Computer Vision

This thesis was completed at:
Institut de Robòtica i Informàtica Industrial, CSIC-UPC

Thesis advisors:
Francesc Moreno Noguer and Alberto Sanfeliu Cortés

Dissertation Committee:
Kalyan Sunkavalli
Xavier Giró Nieto
Gül Varol

BRIDGING THE GAP BETWEEN RECONSTRUCTION AND SYNTHESIS

Albert Pumarola Peris

Abstract

3D reconstruction and image synthesis are two of the main pillars in computer vision. Early works focused on simple tasks such as multi-view reconstruction and texture synthesis. With the spur of Deep Learning, the field has rapidly progressed, making it possible to achieve more complex and high level tasks. For example, the 3D reconstruction results of traditional multi-view approaches are currently obtained with single view methods. Similarly, early pattern based texture synthesis works have resulted in techniques that allow generating novel high-resolution images.

In this thesis we have developed a hierarchy of tools that cover all these range of problems, lying at the intersection of computer vision, graphics and machine learning. We tackle the problem of 3D reconstruction and synthesis in the wild. Importantly, we advocate for a paradigm in which not everything should be learned. Instead of applying Deep Learning naively we propose novel representations, layers and architectures that directly embed prior 3D geometric knowledge for the task of 3D reconstruction and synthesis. We apply these techniques to problems including scene/person reconstruction and photo-realistic rendering. We first address methods to reconstruct a scene and the clothed people in it while estimating the camera position. Then, we tackle image and video synthesis for clothed people in the wild. Finally, we bridge the gap between reconstruction and synthesis under the umbrella of a unique novel formulation. Extensive experiments conducted along this thesis show that the proposed techniques improve the performance of Deep Learning models in terms of the quality of the reconstructed 3D shapes / synthesised images, while reducing the amount of supervision and training data required to train them.

In summary, we provide a variety of low, mid and high level algorithms that can be used to incorporate prior knowledge into different stages of the Deep Learning pipeline and improve performance in tasks of 3D reconstruction and image synthesis.

Keywords: 3D reconstruction, image synthesis, deep learning, generative models, human-centric imaging.

Acknowledgements

I would like to thank my supervisors, Dr. Francesc Moreno Noguer and Prof. Alberto Sanfeliu Cortés, for their valuable guidance and support.

I am grateful to Prof. Fernando de la Torre and Prof. Vittorio Ferrari, who did welcome me to work with them during several months in what became two very fruitful research stays.

Finally, I would also like to thank my family for the support provided throughout my entire life and in particular to Berta, for always standing by my side and enduring absences while I was working on this thesis.

This work has been partially supported by:

- EU project AEROARMS - Aerial Robotics Inspection System, 2015-2019. Code: H2020-ICT-2014-1-644271
- National Project ColRobTransp - Robotics Last Mile Transportation in Urban Areas, 2016-2019. Code: DPI2016-78957-R
- Google Research Award - Geometry-aware CNNs for Non-Rigid Reconstruction, 2017-2018
- Amazon Research Award - Geometry-aware 3D Human Body Animation from Still Photos, 2018-2019
- Google Research Award - GANimation3D: Unsupervised 3D Face Animation from Monocular Images, 2019-2020

Contents

Figures

Tables

Acronyms

3DMM 3D Morphable Model. 77, 160, 161

ACD Average Content Distance. xviii, 102, 104–106

ATE Absolute Trajectory Error. xvii, 27, 29

AU Action Unit. xi, 76, 90–92, 94, 95, 97–100, 102, 127

BA Bundle Adjustment. vii, 15, 20–22, 24, 32, 33

BPD Bits Per Dimension. 143, 145

CD Chamfer Distance. xviii, 133, 139, 141, 143

CNN Convolutional Neural Network. 2, 4, 5, 8, 16, 18, 57, 59, 63, 136

CPU Central Processing Unit. 15

DL Deep Learning. 1–3, 5, 7, 8, 16, 34, 35, 46, 48, 56, 57, 77, 78, 130

ED Expression Distance. xviii, 105, 106

EMD Earth Mover Distance. 11, 38, 39, 95

FACS Facial Action Coding System. 90, 91

FPS Frames per Second. viii, xvii, 29, 33–35, 41–44, 73, 98

GAN Generative Adversarial Network. 10–12, 17, 35, 39, 73, 75–80, 85, 86, 89, 91, 94, 102, 110, 111, 120, 123, 127, 130–132, 136, 145

GIM Geometry Image. ix, x, xvii, 14, 58, 59, 62–72

GPU Graphics Processing Units. 8, 15, 40, 44, 45, 98, 113, 140

IS Inception Score. xviii, 85, 86, 102, 104, 118, 120, 143, 145

JS Jensen-Shannon. 11, 39, 94, 95

LSTM Long Short-Term Memory. 17, 34

MoCap Motion Capture Data. 58, 59, 61

NR-ICP Non-Rigid Iterative Closest Point. 65, 69, 70

PnP Perspective-n-Point. 24

RAM Random-Access Memory. 27, 40, 98

ReLU Rectified Linear Activation Units. 8, 69, 118, 136

RNN Recurrent Neural Network. 17, 37

SfT Shape-from-Template. 46

SLAM Simultaneous Localization and Mapping. 13, 15, 16, 19–21, 24–27, 33, 73, 159

SMPL Skinned Multi-Person Linear. ix, 18, 60, 71, 160

SSIM Structural Similarity. xviii, 85, 86, 145

SVM Support Vector Machine. 1, 7, 8

VAE Variational Auto-Encoder. 10, 12, 77, 85, 130, 132

Nomenclature

Numbers and Arrays

a	Scalar
$\mathbf{a}$	Vector
$\mathbf{A}$	Matrix
$\boldsymbol{a}$	Scalar random variable
$\mathrm{diag}(\mathbf{a})$	Square diagonal matrix with diagonal entries given by $\mathbf{a}$
$\mathbf{t}$	Translation vector
$\mathbf{R}$	Rotation matrix
$\boldsymbol{I}$	Identity matrix
$\mathbf{0}$	Zero matrix
$\mathbf{I}$	Image

Sets and Graphs

$\mathbb{R}$	Set of real numbers
$\mathcal{U}$	Set of all pixels (u, v) of an image $\mathbf{I}$
$\mathcal{D}$	Set of 1-Lipschitz functions
$\{0, \ldots, n\}$	Set of all integers between 0 and n
$[a, b]$	Real interval including a and b

Indexing

a_i	Element i of vector $\mathbf{a}$
$\mathbf{a}_{i:j}$	All elements between the i-th and $(j-1)$-th positions of vector $\mathbf{a}$
$\mathbf{A}_{i,j}$	Element i, j of matrix $\mathbf{A}$
$\mathbf{A}[u, v]$	Element u, v of matrix $\mathbf{A}$

Calculus

$\frac{\partial y}{\partial x}$	Partial derivative of y with respect to x
$\nabla_{\mathbf{x}}$	Gradient of the model parameters with respect $\mathbf{x}$
$\frac{\partial f}{\partial \mathbf{x}}$	Jacobian matrix $\mathbf{J} \in \mathbb{R}^{m \times n}$ of $f : \mathbb{R}^n \to \mathbb{R}^m$

Probability and Information Theory

$\mathbb{P}(\mathrm{a})$	Probability distribution over a discrete variable
$p(\mathrm{a})$	Probability density function over a continuous variable
$\mathrm{a} \sim \mathbb{P}$	Random variable a has distribution $\mathbb{P}$
$\mathbb{E}_{\mathbf{x} \sim P}[f(x)]$	Expectation of $f(x)$ with respect to $P(\mathbf{x})$

$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian distribution over $\boldsymbol{x}$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$)

Functions

$f : \mathbb{A} \to \mathbb{B}$	The function f with domain $\mathbb{A}$ and range $\mathbb{B}$
$f \circ g$	Composition of the functions f and g
$f(\boldsymbol{x}; \boldsymbol{\theta})$ or $f_{\boldsymbol{\theta}}(\boldsymbol{x})$	A function of $\boldsymbol{x}$ parametrized by $\boldsymbol{\theta}$ (Sometimes we omit $\boldsymbol{\theta}$ to simplify notation)
$\log x$	Natural logarithm of x
G	Generator function
D	Discriminator function

Datasets and Distributions

$\mathbb{P}_{data}$	Data distribution
$\boldsymbol{x}^{(i)}$	The i-th sample of a dataset

1 Introduction

Deep Learning (DL) has gained much popularity over the last years and achieved performance breakthroughs in a wide range of applications. DL represents a paradigm change on how we think about machine learning and has revolutionized fields such as natural language processing, robotics, computer graphics, and in particular, computer vision. Previous standard pipelines relied on highly engineered handmade problem-dependant methodologies and feature design techniques that were combined with Machine Learning classifiers, typically Support Vector Machines (SVMs). In contrast, Deep Learning offers a general framework for training end-to-end systems, from raw data, without any feature engineering involved. A chained cascade of non-linear transformations allows the networks to learn powerful image feature representations from large collections of examples which outperform all traditional engineered features.

Nevertheless, this thesis advocates for a paradigm in which not everything should be learned. For most DL-based applications there exist a set of predefined properties and constraints that need to be satisfied, and that should not be learned by the network. Instead, these properties/prior knowledge should be directly embedded into the data representation or as an inherent part of the network architecture. An example where prior knowledge can be greatly exploited is in modeling the distribution of 3D objects, that is, to learn the manifold of plausible objects in terms of their 3D shape and visual appearance. To learn such distribution, one should exploit well established computer vision and graphics techniques such as camera projection models, efficient 3D data representations and volume rendering.

Being able to model the distribution of 3D objects would open the door to many new exciting applications in different areas, including 3D reconstruction and neural rendering, key components for scene understanding and augmented reality. Recent advances in DL and generative models have shown impressive results for the related task of 2D image modeling and synthesiszing photo-realistic images, face generation and super-resolution imaging. However, these methods are not able to escalate to 3D data and become intractable in terms of space and

time complexity making it, in practice, intractable. Moreover, convolutional based methods such as deep Convolutional Neural Network (CNN) are not well fitted for 3D representation and the community has not yet agreed on a 3D object representation that is both memory efficient and easily modeled by learning methods.

This thesis lies at the intersection of computer vision, graphics, and machine learning. We tackle the problem of 3D reconstruction and 2D synthesis for the applications of scene/person reconstruction and photo-realistic rendering. Importantly, we do not apply Deep Learning naively with fully Convolutional Neural Networks, but instead with novel representations, layers and architectures which embed prior knowledge of the problem. For example, we propose a fully differentiable layer that models the pin-hole projection model without learning parameters and a new representation for 3D meshes well fitted for CNNs. We believe that it is only by enhancing networks with this knowledge that they will achieve super-human performance in computer vision problems. And most importantly, we pursue models that can be trained without requiring vast amounts of training data, and, in some scenarios, without ground-truth annotations.

The methods developed in this thesis have contributed and been implemented in several competitive projects: (i) Real-time scene reconstruction and precise localization of UAV vehicles in the EU project "AEROARMS" H2020-ICT-2014-1-644271; (ii) Non-rigid 3D shape estimation from a single image in the 2017 *Google Faculty Research Award* "Geometry-aware CNNs for Non-Rigid Shape Reconstruction"; (iii) 3D reconstruction of human body and cloth from a single image in the 2019 *Amazon Research Award* "Geometry-aware 3D Human Body Animation from Still Photos"; (iv) Unsupervised baseline method in which to integrate a 3D geometry model for facial expressions rendering in the 2019 *Google Faculty Research Award* "GANimation3D: Unsupervised 3D Face Animation from Monocular Images".

It is also worth mentioning that GANimation [18], the face animation algorithm, developed in this thesis received the "Best Paper Honorable Mention" award at ECCV 2018 out of 2439 valid submissions.

1.1 Objectives

The main goal of this thesis is to incorporate *prior knowledge* into different stages of the Deep Learning pipeline and improve *performance* in tasks of 3D reconstruction and image synthesis.

Let us clarify two keywords of this statement: 'prior knowledge' and 'performance'. As a 'prior knowledge' we consider different types of data representation that can be easily processed by the networks, and novel layers/architectures that inherently reason about the geometric and physical constraints of the problem. Additionally, although 'performance' is a quite generic concept, here we measure it in terms of the quality of the reconstructed 3D shapes or synthesised

images; in terms of the amount of training data that is required; and in terms of the supervision that it needed.

We next summarize the specific goals of our research:

- Design novel data structures to represent 3D objects that are both memory efficient and easily modeled by DL architectures.
- Build novel DL-based models and loss functions that enforce 3D geometric constraints.
- Create new realistic and diverse benchmarks for the comparison and evaluation of reconstruction methods.
- Apply the above goals to 3D reconstruction and synthesis in the wild.

1.2 Contributions

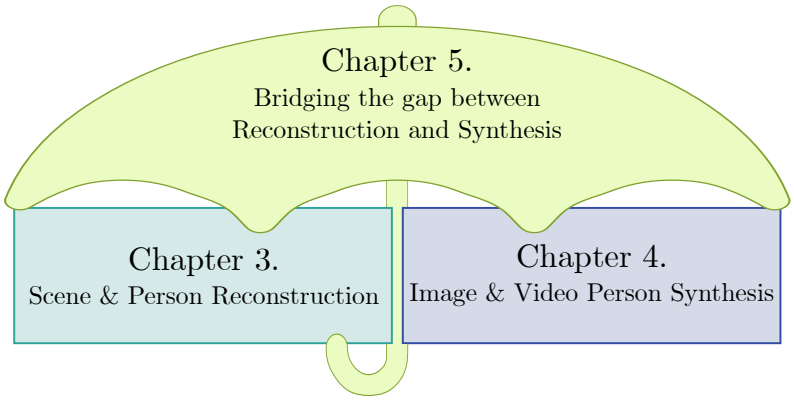


Figure 1.1: **Thesis overview**.

The contributions of this dissertation are organized in three main blocks: reconstruction, synthesis and its conjunction (Fig. 1.1). In the first block, we present a pipeline for simulateous reconstruction of scenes / clothed people, and camera pose estimation. In the second block, we further model 3D data, but this time through novel generative methods to model the nonrigid articulated human body. Finally, in the last block, we bridge the gap between reconstruction and synthesis with new general-purpose models. We next list the main contributions for each block.

Scene & Person Reconstruction Contributions:

1. We introduce an algorithm for scene 3D reconstruction and camera pose estimation robust to poorly textured scenarios that builds upon the joint estimation of point and line correspondences [19–21].

2. We present a method to learn spatio-temporal object models for real-time detection and tracking from a single reference image [22]. This was the first approach able to run at real-time.

3. Build DeformNet [23], a large dataset and benchmark of synthetic renderings of non-rigid surfaces under different levels of deformation, material properties, textures and lighting conditions.

4. Propose a novel architecture for predicting the 3D shape of a deformable surface from a single view [23]. By contrast with previous approaches, it does not require a pre-registered template of the surface, and is robust to the lack of texture and partial occlusions.

5. Build 3DPeople [24], the first large-scale synthetic dataset and benchmark with 2.5 Million photo-realistic images of 80 subjects performing 70 activities and wearing diverse 3D outfits. Besides providing textured 3D meshes for clothes and body, we annotate the dataset with segmentation masks, skeletons, depth, normal maps and optical flow. All this together makes 3DPeople suitable for a plethora of tasks.

6. Propose a representation for 3D meshes [24] that maps 3D data into a 2D space emulating an image, therefore being a 3D data representation compatible with all existing literature on image-based CNN architectures.

7. Build a model that, given an input image of a dressed human in the wild, reconstructs the 3D mesh of the body pose and clothing shape [24].

Image & Video Person Synthesis:

8. Design the first unsupervised method to synthesize photo-realistic images of people in arbitrary poses [25]. Given an input image of a person and a desired pose represented by a 2D skeleton, the model renders the image of the same person under the new pose, synthesizing novel views of the parts visible in the input image and hallucinating the non-visible.

9. Propose an unsupervised method that allows facial expression synthesis [18, 26]. Previous approaches were only suitable for editing facial expressions out of a discrete set of emotion categories. Ours was the first method able to animate facial expression in a continuous manner.

10. Develop plug-and-play attention mechanisms for image editing networks to make them robust to changing backgrounds and lighting conditions in images in the wild [18, 26].

11. Devise the first unsupervised system to photo-realistically transfer the clothing of a person in a reference image into another person in an unconstrained image or video in the wild [27].

12. Build a novel plug-and-play CNN layer mimicking a physical memory for consistent video frames synthesis [27].

Bridging the gap between Reconstruction & Synthesis:

13. Develop a novel formulation for conditioning flow models [28]. We demonstrate our conditioning method to be very adaptable, being applicable to 3D reconstruction, image synthesis and manipulation, style transfer and multi-modal mapping in a diversity of domains, including RGB images, 3D point clouds, segmentation maps, and edge masks.

14. Present a novel bijective plug-and-play CNN layer to encode 3D point clouds for bijective architectures [28].

15. Introduce a method able to reconstruct and render novel images of objects under rigid and non-rigid motions given a sparse set of images captured by a single camera moving around the scene [29].

For a detailed *list of publications* derived from this thesis refer to Appendix A.

1.3 Outline

This thesis is organized in six chapters. The first chapter introduces, motivates and defines its initial goals and final contributions. The second chapter establishes groundwork for the techniques used. The following three chapters explain the contributions of the thesis, which are conceptually grouped into the three blocks aforementioned: reconstruction, synthesis and its unification (Fig. 1.1). Each of these chapters has an introduction to the topic, discussion of the previous work, several sections explaining the developed techniques, each corresponding roughly to a single publication, and a final summary of the work. Finally, the last chapter provides the closing remarks of this thesis.

- **Chapter 1: Introduction**. The first chapter introduces the motivation, the main objectives and an overview of the contributions done throughout the thesis.

- **Chapter 2: Overview**. Establishes groundwork for the techniques and tools used throughout the thesis. It is divided in: Deep Learning, learning methods and generative models.

- **Chapter 3: Scene & Person Reconstruction**. Chapter devoted to explain the proposed methods for reconstructing a scene and the clothed people in it while estimating the camera position. Eight publications are related to this chapter: five already published [19–21, 23, 24] and three under revision.

- **Chapter 4: Image & Video Person Synthesis**. Introduces three unsupervised methods for clothed people synthesis and edition in the wild. Five publications are related to this chapter: four already published [18, 25–27] and one under revision.

- **Chapter 5: Bridging the gap between Reconstruction and Synthesis**. This chapter combines the previous chapters, bringing reconstruction and synthesis under the umbrella of two novel formulations: [28] for generalization to large datasets and [29] to model one specific scene at a time. Two publications are related to this chapter: [28, 29].

- **Chapter 6: Closing Remarks**. The last chapter summarizes and discusses all our contributions and conclusions. It also gives a high-level description of future research lines.

2
Overview

The goal of this chapter is to give the reader a global perspective of the elements common to the techniques used in this dissertation. We first introduce Deep Learning, the common denominator of most of our models (Sec. 2.1). Then, we discuss the different learning paradigms we have considered (Sec. 2.2). Finally, we introduce and explain the required concepts on generative models (Sec. 2.3). Table 2.1 summarises the reviewed techniques and how they relate to each work presented in this dissertation.

2.1 Deep Learning

Machine learning deals with the problem of learning from data and the development of algorithms capable of tackling a specific task by relying on patterns from a training dataset instead of hard-coded instructions. Machine learning plays a fundamental role in modern computer vision. There is a vast family of techniques in the literature such as Support Vector Machine [30], decision trees and conditional random fields [31]. However, the best performing solutions for computer vision nowadays are mostly based on Deep Learning (Deep Learning).

Over the past 8 years, DL has grown in popularity thanks to its performance breakthroughs in works in a wide range of applications. The first neural network dates back to 1943 when Warren McCulloch and Walter Pitts introduced the concept of Thresholded Logic Unit as a model of how a human neuron was thought to work [32]. In the 1958 Frank Rosenblatt proposed Mark I Perceptron, the first precursor to modern nets. An end-to-end system based on the McCulloch-Pitts neuron capable of iteratively optimizing the perceptron weights by minimizing the difference between desired and predicted output. This was considered to be key to true pure intelligence.

Minsky destroyed this idyllic believe by proving that a perceptron could not even learn the XOR function, and could only discriminate between linearly separable classes [33]. Also it

		DL	Sup.	Unsup.	Semi-sup.	Gen.
Chapter 3	Sec. 3.3 - Scene Reconstruction and Camera Pose Estimation					
	Sec. 3.4 - Object Detection and Tracking	✓			✓	✓
	Sec. 3.5 - Non-Rigid Surface Reconstruction	✓	✓			
	Sec. 3.6 - Modeling the Geometry of Dressed Humans	✓	✓			✓
Chapter 4	Sec. 4.3 - Person Image Synthesis	✓		✓		✓
	Sec. 4.4 - Face Image Synthesis and Animation	✓		✓		✓
	Sec. 4.5 - Image-to-Video Cloth Transfer	✓		✓		✓
Chapter 5	Bridging the gap between Reconstruction and Synthesis	✓	✓			✓

Table 2.1: **Overview.** Index of the overviewed techniques and how they relate to the methods presented in this thesis. 'DL' stands for Deep Learning (Sec. 2.1). 'Sup.', 'Unsup.' and 'Semi-sup.' stand for Supervised, Unsupervised and Semi-supervised (Sec. 2.2). 'Gen.' stands for Generative (Sec. 2.3).

was argued that multi-layered neural networks would take infinite amount of time to learn its weights. Nevertheless, in 1986, Geoff Hinton *et al.* [34] demonstrated that multi-layer networks could be trained by a simple method based on back-propagating the error across the network [34]. Backpropagation and the Universal Approximation Theorem stating that neural networks were potential universal approximators lead to the appearance of Convolutional Neural Networks [35]. But its lack of scalability made neural nets not competitive against SVM [30]. It was not until 2012 that neural networks regain popularity when Alex Krizhevsky *et al.* won the ImageNet [36] competition with the convolutional network architecture AlexNet [37] introducing key improvements to the original CNNs as Dropout, Rectified Linear Activation Units (ReLU) and training the model with Graphics Processing Unitss (GPUs). The latter allowed training with large amounts of data in reasonable amounts of time.

At its very core, neural networks, are formally defined as a directed acyclic graphical model consisting of a compositional function in which the nodes represent non-linear transformations:

$$f(\mathbf{x};\boldsymbol{\theta}) = W_L\sigma_L(W_{L-1}\cdots\sigma_2(W_2\sigma_1(W_1\mathbf{x}))) \tag{2.1}$$

where $\boldsymbol{\theta} = \{W_1, ..., W_L\}$ are the parameters of a L-layers network and every $\sigma_l(\cdot)$ is a nonlinear activation function. The parameters $\boldsymbol{\theta}$ are learned using gradient descent. More adaptive optimization algorithms like RMSProp and Adam [38] are also commonly used in Deep Learning. Throughout this thesis we used Adam.

While traditional machine learning algorithms have great performance on small training datasets, for large datasets deep neural networks outperform the rest of methods for two main reasons: (i) Large neural networks have many more parameters than traditional approaches,

making it possible to learn complex nonlinear patterns from data. (ii) Deep neural networks do not require hand-crafted features, they are able to extract low and high-level features directly from raw data.

2.2 Supervised vs. Unsupervised vs. Semi-supervised Learning

There exist several taxonomies to classify machine learning models. Here we consider a classification based on the level of supervision during training. Three main categories are defined, namely: supervised, unsupervised and semi-supervised. We next define and discuss their implications in this thesis.

Supervised Learning. Learning procedure that consists on learning from labeled data with training pairs of input and desired output. For example, this would consist in learning a cat breed classifier using a dataset containing diverse images of cats each labeled with its corresponding breed class. Supervised learning is the most commonly used and more stable to train among all learning methods. Supervision allows to be very specific with the expected behavior of the model, as one can define the exact desired output by penalizing the difference between the desired and regressed output. The main pitfall of supervised learning is having to go through the expensive procedure of ground-truth annotation. In some cases, setting a label per sample can be challenging to define, requiring the use of experts (*e.g.* medical data). It can even be close to impossible to set (*e.g.* annotating the time varying shape of a deformable object given a single image).

Unsupervised Learning. When considering an unsupervised learning strategy the input training data is provided with no corresponding ground-truth output. In other words, the algorithm is expected to find by its own the hidden patterns and structure in unlabeled data. Let us consider a classification algorithm trained to cluster images of cats. The classification would be based on the hidden patterns found in the data and, not necessarily correlated with the human concept of 'breed'. Unsupervised learning allows to tackle tasks for which humans do not have a specific answer, finding hidden patterns and structures in the data than can determine unexplored solutions. The main advantage of unsupervised learning is the ease of gathering training data, as finding unlabeled data is easier than finding labeled, this last being necessary for supervised methods.

Semi-supervised Learning. This learning paradigm can be understood as a combination of supervised and unsupervised learning in which the data is mostly unlabeled and only some of it is labeled. In the previous example of classifying cat breeds, this would consist in using a training dataset containing diverse images of cats but only some of the images are labeled with its corresponding breed. However, although semi-supervised learning apparently seems to get the best of both supervised and unsupervised learning, it cannot provide significant advantages over supervised learning unless there is some non-trivial relationship between labels and the unlabeled data.

Which is the best learning strategy? There is no universal answer, the selection of the learning algorithm depends entirely on the task use cases and the available data.

We live in a society generating ~ 40 zettabytes (10^{21} bytes) of data daily. Meaning that there is plenty of publicly available but unlabeled training data and it is impossible to keep up to date with manual annotations. The authors opinion is that the future is unsupervised, and more precisely its subcategory **self-supervised**. Self-supervised learning aims to use large amounts of unlabeled data by defining learning objectives supervised by the data itself. In other words, instead of learning from manually annotated data, learn by using some part of the inputs or transformed inputs as targets.

Across this thesis we leverage on each of the previous learning strategies (see Table 2.1). For instance, in Chapter 3 we rely on a fully supervised approach to reconstruct the geometry of deformable surfaces. This same chapter also explores a semi-supervised video segmentation and tracking algorithm which, in test, can be applied to previously unknown objects. In Chapter 4, the proposed synthesis methods are trained in a self-supervised manner, as they do not require direct supervision nor ground-truth on the specific desired task. Finally, Chapter 5 sets a solid supervised baseline for novel solutions for reconstruction and synthesis.

2.3 Generative Models

Generative models aim to approximate an unknown true data distribution $\mathbf{x} \sim p^*(\mathbf{x})$ from a limited set of observations $\{\mathbf{x}^{(i)}\}_{i=1}^N$ by generating new synthetic data similar to the true data. The generated data is intended to be novel, and not copies of the original data. To prevent the model from acting as a copying hash table, the number of parameters of generative models needs to be significantly smaller than the size of the training set. By doing so, the model is forced to discover the data essence at its very core in order to then generate it.

The most popular generative models are the Variational Auto-Encoders (VAEs) [39] and Generative Adversarial Networks (GANs) [40]. VAEs maximize a lower bound on the data log-

likelihood to model a continuous latent variable with intractable posterior distribution. GANs, on the other hand, circumvent the need of dealing with likelihood estimation by leveraging a min-max game. More recently flow-based generative models [41, 42] have started to gain popularity, being the only generative models that allow explicitly estimating the exact data probability density function.

We next explain in more detail the two generative methods explored throughout this thesis: Generative Adversarial Network and flow-based generative models.

Generative Adversarial Networks (GANs)

GANs are a class of generative models based on game theory that, when applied to image generation, have been shown to produce very realistic results with a high level of detail. The original GAN formulation is based on the Jensen-Shannon (JS) divergence, consisting in simultaneously training a generator G to produce realistic fake samples and a discriminator D trained to distinguish between the real data from the true distribution and the fake data from the approximated distribution. This idea is embedded into the so-called *adversarial loss*:

$$\min_{G} \max_{D} \mathbb{E}_{\mathbf{z}\sim\mathbb{P}_z}[\log(1 - D(G(\mathbf{z})))] + \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_x}\left[\log D(\mathbf{x})\right], \tag{2.2}$$

where $\mathbf{z}$ is random noise, $\mathbb{P}_z$ the Gaussian distribution $\mathcal{N}(\mathbf{z}; \mathbf{0}, \boldsymbol{I})$, $\boldsymbol{I}$ the identity matrix, $\mathbf{x}$ is the input data and $\mathbb{P}_x$ the input data distribution. This loss is potentially not continuous with respect to the generator's parameters and can locally saturate leading to vanishing gradients in the discriminator. Recent works [43, 44] have shown improved stability by replacing JS with the continuous Earth Mover Distance (EMD) metric. In practice, computing the EMD is instractable. Using the Kantorovich-Rubinstein duality it can be approximatly estimated as long as the Discriminator is constrained to lie on the set of 1-Lipschitz functions. To maintain a Lipschitz constraint $\mathcal{D}$, WGAN-GP [44] proposes to add a gradient penalty computed as the norm of the gradients with respect to the discriminator input.

Formally, let $\widetilde{\mathbf{x}}$ be a randomly weighted linear interpolation between the real data $\mathbf{x}$ and the generated data $G(\mathbf{x})$, $\mathbb{P}_{\widetilde{x}}$ the random interpolation distribution and λ_{gp} the weight of the gradient penalty regularization. Then, the new loss is defined as:

$$\min_{G} \max_{D\in\mathcal{D}} -\mathbb{E}_{\mathbf{z}\sim\mathbb{P}_z}[D(G(\mathbf{z}))] + \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_x}\left[D(\mathbf{x})\right] - \lambda_{\text{gp}}\mathbb{E}_{\widetilde{x}\sim\mathbb{P}_{\widetilde{x}}}\left[(\|\nabla_{\widetilde{x}} D(\widetilde{\mathbf{x}})\|_2 - 1)^2\right] \tag{2.3}$$

An active research area consists in designing GAN models that incorporate conditions and constraints into the generation process. Prior studies have explored combining several conditions, such as textual descriptions [45, 46] and class information [47, 48]. Particularly in-

teresting for this thesis are those methods exploring image-based conditioning as in image super-resolution [49], future frame prediction [50], image in-painting [51], image-to-image translation [52] and multi-target domain transfer [11].

Flow-Based Generative Models

In Flow-based generative models the data is modeled by learning an invertible transformation $\mathbf{g}_{\boldsymbol{\theta}}(\cdot)$ mapping from a latent space with tractable density $p_{\boldsymbol{\vartheta}}(\mathbf{z})$ to the true data $\mathbf{x}$:

$$\mathbf{z} \sim p_{\boldsymbol{\vartheta}}(\mathbf{z}), \quad \mathbf{x} = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z}), \tag{2.4}$$

where $\mathbf{z}$ is a latent variable and $p_{\boldsymbol{\vartheta}}(\mathbf{z})$ is typically a Gaussian distribution $\mathcal{N}(\mathbf{z}; \mathbf{0}, \boldsymbol{I})$. The function $\mathbf{g}_{\boldsymbol{\theta}}$, commonly known as normalizing flow [42], is bijective, meaning that given a data point $\mathbf{x}$ its latent-variable $\mathbf{z}$ is computed as:

$$\mathbf{z} = \mathbf{g}_{\boldsymbol{\theta}}^{-1}(\mathbf{x}), \tag{2.5}$$

where $\mathbf{g}_{\boldsymbol{\theta}}^{-1}$ is composed of a sequence of K invertible transformations $\mathbf{g}^{-1} = \mathbf{g}_1^{-1} \circ \mathbf{g}_2^{-1} \circ \cdots \circ \mathbf{g}_K^{-1}$ defining a mapping between $\mathbf{x}$ and $\mathbf{z}$, such that:

$$\mathbf{x} \triangleq \mathbf{h}_0 \overset{\mathbf{g}_1^{-1}}{\longleftrightarrow} \mathbf{h}_1 \overset{\mathbf{g}_2^{-1}}{\longleftrightarrow} \mathbf{h}_2 \cdots \overset{\mathbf{g}_K^{-1}}{\longleftrightarrow} \mathbf{h}_K \triangleq \mathbf{z}, \tag{2.6}$$

being K a fixed hyper-parameter.

The goal of generative models is to find the parameters $\boldsymbol{\theta}$ such that $p_{\boldsymbol{\theta}}(\mathbf{x})$ best approximates the true data distribution $p^*(\mathbf{x})$. Explicitly modeling such probability density function is usually intractable, but using the normalizing flow mapping of Eq. (2.4) under the change of variable theorem, we can compute the exact log-likelihood for a given data point $\mathbf{x}$ as:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \log p_{\boldsymbol{\vartheta}}(\mathbf{z}) + \log |\det(\partial \mathbf{z}/\partial \mathbf{x})| \tag{2.7}$$

$$= \log p_{\boldsymbol{\vartheta}}(\mathbf{z}) + \sum_{i=1}^{K} \log |\det(\partial \mathbf{h}_i/\partial \mathbf{h}_{i-1})| \tag{2.8}$$

where $\partial \mathbf{h}_i/\partial \mathbf{h}_{i-1}$ is the Jacobian matrix of $\mathbf{g}_i^{-1}$ at $\mathbf{h}_{i-1}$, and the Jacobian determinant measures the change of log-density made by $\mathbf{g}_i^{-1}$ when transforming $\mathbf{h}_{i-1}$ to $\mathbf{h}_i$. Since we can now compute the exact log-likelihood, the training criterion of flow-based generative model is directly the negative log-likelihood over the observations. Note that optimizing over the actual log-likelihood of the observations is more stable and informative than doing it over a lower-bound of the log-likelihood for VAEs, or minimizing the adversarial loss in GANs. This is one of the major virtues of flow-based approaches.

3

Scene & Person Reconstruction

3.1 Introduction

3D reconstruction methods aim to reconstruct a scene and the objects in it from a single view, have the potential to open the door to many applications in augmented reality and scene understanding. This is an extremely challenging task. Taking a picture is equivalent to projecting the 3D world to a 2D image plane losing depth information. Trying to recover depth is known to be a severely ill-posed problem, and it requires introducing different sources of prior knowledge in order to be solved. Another problematic are non-visible areas, objects are commonly highly occluded and/or have self-occlusions. Reconstruction under occlusions is particularly challenging due to the inherent ambiguity that exists in the reconstructions of non-visible parts. This chapter is devoted to explain four novel methods that are tailored to reconstruct a scene and/or the clothed persons in it, while estimating the camera position. All the proposed methods are monocular and propose strategies to handle occlusions, self-occlusions and low-textured surfaces.

First, we present a method for scene reconstruction and camera localization specially designed for the challenging case of low textured scenes where textured key-points are scarce (Sec. 3.3). Yet, there are many environments in which, despite being low textured, one can still reliably estimate line-based geometric primitives, for instance in city and indoor scenes, or in the so-called *manhattan worlds*, where structured edges are predominant. We present a Simultaneous Localization and Mapping (SLAM) based method that relies on the edges and lines of the scene as geometric primitives. We thoroughly benchmark the proposed approach and demonstrate that the use of lines does not only bost the performance of the state-of-the-art solution in poorly textured frames, but also systematically improves it in sequences combining points and lines, without compromising the efficiency.

Second, we present a spatio-temporal method to learn object models for real-time object segmentation and tracking from a single reference image (Sec. 3.4). To achieve this, we concentrate all the heavy computational load to the training phase with two generative model discriminators that enforce spatial and temporal mask consistency over the last K frames. Then at test time, we only use a light regressor, which considerably reduces the inference time. As a result, the final model combines a high resiliency to sudden geometric and photometric object changes with efficiency at test time. We demonstrate its accuracy to be on par with the state-of-the-art techniques on the challenging benchmarks, while running at 32 fps, about 4x faster than the closest competitor. This methods was the first to run inference at real-time for the specific task of object segmentation and tracking from a single reference image.

Once we can segment objects in a scene, we then explore the problem reconstructing them. As a third contribution, we propose a method for estimating the 3D shape of a deformable surfaces with simple topologies (rectangular) from a single view (Sec. 3.5). In contrast to previous approaches, our technique does not require a pre-registered template of the surface, and is robust to the lack of texture and partial occlusions. The core of this approach is a geometry-aware deep architecture that tackles the problem as usually done in previous non-deep learning solutions: first perform 2D detection of the mesh and then estimate a 3D shape that is geometrically consistent with the image. We also build a large dataset and benchmark of synthetic renderings of shapes under different levels of deformation, material properties, textures and lighting conditions. We benchmark the approach and demonstrate it consistently improves the state-of-the-art solutions with a significantly lower computational time, about 67x faster than the closest competitor in performance.

Finally, once we are capable of detecting objects in a scene and performing monocular reconstruction of surfaces with simple topologies, we tackle the more complex task of monocular reconstruction of dressed humans (Sec. 3.6). We contribute in three fundamental aspects of the problem, namely, a new dataset, a novel shape parameterization algorithm and an end-to-end deep generative network for predicting shape. First, we present 3DPeople, a large-scale synthetic dataset with 2.5 Million photo-realistic images of 80 subjects performing 70 activities and wearing diverse outfits. Besides providing textured 3D meshes for clothes and body, we annotate the dataset with segmentation masks, skeletons, depth, normal maps and optical flow. All this together makes 3DPeople suitable for a plethora of tasks. We then introduce a new 3D shapes representation based on 2D Geometry Images (GIMs). Lastly, we design a multi-resolution deep generative network that, given an input image of a dressed human, predicts his/her GIMs (and thus the clothed body shape) in an end-to-end manner. We obtain very promising results in jointly capturing body pose and clothing shape, both for synthetic validation and on the wild images.

3.2 Related Work

We next elaborate on the state-of-the-art of all necessary aspects required to reconstruct a scene and the clothed persons in it while estimating the camera position: scene reconstruction, camera pose estimation, object segmentation and object reconstruction.

Scene Reconstruction and Camera Pose Estimation

Building the 3D rigid structure of unknown environment while recovering the camera trajectory from a monocular image sequence has been a main research area in robotics and computer vision for decades, with many real applications in autonomous robot navigation and augmented reality. This problem is known as Simultaneous Localization and Mapping (SLAM), and its core is roughly the same compared to structure-from-motion algorithms.

Early filtering approaches applied the Extended Kalman Filter [53] to process every frame in the video for small maps, providing sparse but real-time solutions. Subsequent works based on Bundle Adjustment (BA) handled denser maps just using key-frames to estimate the map [54, 55], obtaining more accurate solutions [56] than filtering techniques. Most approaches rely on the Parallel Tracking and Mapping (PTAM) algorithm [54], that represented a breakthrough in visual-based SLAM. This method approximately decouples localization and mapping in two threads that run in parallel, relying on FAST [57] corners points. In [58] the accuracy was improved with edge features together with a rotation estimation step during tracking that provided better relocalization results, and even reducing the computational cost [59]. More recently, the ORB-SLAM system has been proposed in [1], providing a more robust camera tracking and mapping estimator. A multi-threaded Central Processing Unit (CPU) approach was presented in [60] to estimate real-time dense structure estimation.

However, all previous feature-based methods fail in environments with poor texture or situations with defocus and motion blur. To solve this, dense and direct methods can be applied, even though they are likely to be computationally expensive [61, 62], and require dedicated GPU-implementations to achieve real-time performance. Other semi-direct methods such as [63] overcome the high-computation requirement of dense methods by exploiting only pixels with strong gradients, providing an intermediate level of accuracy, density and complexity. Scene prior information has been also exploited to provide a significant boost to SLAM systems [64,65].

Motivated by the need for efficient and accurate scene representations even for poorly textured environments, in Sec. 3.3 we propose a novel visual-based SLAM system that can combine points and lines information in a unified framework while keeping the computational cost. Note that several parametrizations to combine points and lines were used in EKF-SLAM [66].

However, as we said above, filtering-based approaches have been outperformed by optimization-based approaches in rigid SLAM, as we do in this thesis. We validate our method on a wide variety of scenarios, outperforming state-of-the-art solutions for highly textured sequences and showing very accurate solutions in low-textured scenarios where standard feature-based methods fail.

Object Segmentation

In recent years, object segmentation and tracking have experienced a tremendous increase in popularity due to the publication of large datasets (DAVIS [67, 68] and YouTube-VOS [69]) that have enabled the training of Deep Learning techniques. The two main settings to tackle this problem are semi-supervised and unsupervised. In the former, the ground truth mask for the object of interest in the first frame of the video sequence is given to the method whereas in the latter no information is given to the algorithm and usually the object with predominant motion is segmented. In this thesis, we tackle the semi-supervised setting.

For semi-supervised segmentation, traditional approaches used temporal super-pixel [70, 71], optimization in the bilateral space [72], or optimal selection of object proposals [73] to obtain the object segmentation mask for each frame in the video sequence. [74] and [75] were the first two approaches to apply Deep Learning to the problem. Specifically, [74] fine-tuned the network using the first frame of the video sequence whereas [75] used the mask from the previous frame as an input to the network. [76] extended [74] with an online learning strategy, while [77] also extended [74] by combining its result with Mask-RCNN [78].

Another set of techniques have tried to incorporate the information of the first frame in different ways. [79] formulated the problem as a pattern matching with the initial mask, [80] introduced the initial mask in the network in a batch-norm like layer. [81] used a siamese network to combine the low level features of the initial frame with the current one together with the back propagation through time training strategy introduced in [82].

Moreover, some methods have tried to leverage metric learning to solve the problem [83–86], divide the object in multiple parts and track each of them [87], integrate CNN features with traditional energy minimization techniques [88, 89] or design a complex architecture with re-identification and bidirectional propagation modules [90].

Previous approaches mostly rely on single image segmentation, using at most the previous mask as a temporal consistency constraint. There are, however, a few attempts to exploit the temporal dimension better. For instance, some techniques have leveraged optical flow as an additional input to their network. [91] and [92] built a second CNN branch to process optical flow, [93] used it as a prior in the decoder of the network and, [94] used it to align the features from previous frames. While these methods use optical flow priors trained in a separate context,

[94] used the large scale YouTube-VOS dataset (released by them) to train a convolutional Long Short-Term Memory (LSTM) [95] in an end-to-end manner. The proposed approach in Sec. 3.4 lies in between methods that do not use temporal consistency, and [94, 96], which learns long-term dependencies with an Recurrent Neural Network (RNN).

Close to our approach, several works rely on generative models for semantic segmentation [97–100]. [97] trained the discriminator to differentiate between real and predicted probability maps. [99] used two different discriminators to obtain local and global semantic consistency for the human part segmentation problem. [100] tackled the semi-supervised semantic segmentation task using the discriminator to obtain a confidence label map for unlabeled data. To the best of our knowledge, the segmentation and tracking method we propose in Sec. 3.4 is the first one to successfully apply GANs to video segmentation. Furthermore, we can use a considerably higher image resolution than previous segmentation works (512×512 vs. 256×256 in [99] or 321×321 in [100]) and we leverage recent advances introduced in the images synthesis task, i.e., WGAN with gradient penalty [44], in order to improve stability during training.

Non-Rigid Object Reconstruction

Reconstructing deformable objects from monocular images is known to be a severely ill-posed problem which requires introducing different sources of prior knowledge. We split related work on object reconstruction into methods that define these priors based on pre-defined models (either physically-based or handcrafted) and techniques that learn them from training data. Also, we include a discussion on what is the most appropriate 3D object representation.

Early approaches described non-rigid surfaces using models inspired by physics, such as superquadrics [101], thin-plates [102], elastic models [103] and finite-elements [104]. These representations, however, could not accurately approximate the non-linear behavior of large deformations. More complex deformations can be captured by shape-from-texture approaches [105–114], which aim at recovering the surface geometry given a reference configuration in which the template shape is known, and a set of 3D-to-2D correspondences between this shape and the input image. On top of this, additional constraints enforcing isometry [112], conformal warps [105] and photometric consistency [107, 108] are considered. While effective, shape-from-texture methods are very sensitive to the initial set of matches, which may be difficult to establish in practice, especially under occlusions, low textured surfaces and varying illumination.

Temporal information is another typically exploited prior. Non-rigid-shape-from-motion techniques generally extend Tomasi and Kanade's rigid factorization algorithm [115] to recover deformable shape and camera motion from a sequence of 2D tracks, exploiting physical [116] and low-rank constraints on the shape [117–121], trajectory [122] or the forces inducing the

deformation [123]. Again, these methods rely on the fact that 2D point tracks can be readily computed, limiting thus their general applicability to relatively well-textured surfaces. There exist approaches to describe patches [124] or points of interest [125,126] for deformable scenes, although they are typically computationally expensive [125–127] or require depth information [128].

The need of point correspondences is circumvented by template-free approaches that perform a per-point 3D reconstruction by minimizing an objective function on geometric and photometric cues [129–132]. The shading models considered by these approaches, however, use to be oversimplifications of the reality, either considering brightness constancy [131] or Lambertian surfaces lit by point light sources [132].

More realistic deformation and appearance models can be learned from training data. The first attempt along this line corresponds to the active appearance models [133], which learned low-dimensional 2D models for face tracking. This was later extended to 3D by the active shape and morphable models [134, 135], and by methods integrating these models into the shape-from-texture formulation [136]. Yet, all these approaches for surface reconstruction still rely on feature points detected over the whole surface or at its boundary [137], which are difficult to obtain in practice. In contrast to previous approaches, in Sec. 3.5, we introduce a method that does not require a pre-registered template of the surface and is robust to the lack of texture and partial occlusions.

When dealing with more complex deformations such as human shape estimation, learning methods typically use shape embeddings learned from body scan data repositories like SMPL model [138]. The body geometry is described by a reduced number of pose and shape parameters [139–141]. Dibra *et al.* [142] are the first in using a CNN fed with silhouette images to estimate shape parameters. In [143,144] SMPL body parameters are predicted by incorporating differential renders into the deep network to directly estimate and minimize the error of image features. On top of this, [6] introduces an adversarial loss that penalizes non-realistic body shapes. [145, 146] extended the SMPL parametric representation to model cloth and [147] used shape from shading to predict higher details. Very recently, [148] extended texture maps and [149] implicit functions to regress full body 3D meshes.

Still, which is the most appropriate 3D object representation to train a deep network remains an open question, especially for non-rigid bodies. Standard non-parametric representations for rigid objects include voxels [150, 151], octrees [152–154] and point-clouds [155]. [5, 156] use 2D embeddings computed with geometry images [157] to represent rigid objects. Bodynet [158] explores a network that predicts voxelized human body shape. Very recently, [159] introduces a pipeline that given a single image of a person in frontal position predicts the body silhouette as seen from different views, and then uses a visual hull algorithm to estimate 3D shape. Finally, in

3DPeople [23] (Sec. 3.6) we extend the geometry image representation [157] to accommodate for objects with elongated parts like the human body.

3.3 Scene Reconstruction and Camera Pose Estimation

The last years have witnessed a surge in augmented reality. Among other technologies, at the core of these systems lie sophisticated SLAM algorithms, which have proven effective to accurately reconstruct the unknown environment while estimating the camera position.

Since the groundbreaking Parallel Tracking And Mapping [54] algorithm was introduced by Klein and Murray in 2007, many other real-time visual SLAM approaches have been proposed, including the feature point-based ORB-SLAM [1, 160], and the direct-based methods LSD-SLAM [60] and RGBD-SLAM [161] that directly optimize over image pixels. Among them, the ORB-SLAM seems to be the current state-of-the-art, yielding better accuracy than the direct methods counterparts.

While the performance of ORB-SLAM [1] in well textured sequences is impressive, it is prone to fail when dealing with poorly textured videos or when feature points are temporary vanished out due to, *e.g.*, motion blur. These kind of situations are often encountered in man-made scenarios. However, despite the lack of reliable feature points, these environments may still contain a number of lines that can be used in a similar manner.

Exploiting lines, though, is not a trivial task. First, existing line detectors and parameterizations are not as well-established in the literature as feature point ones. And secondly, the algorithms to compute pose from line correspondences are less reliable than those based on points and are very sensitive to the partial occlusions that lines may undergo. These reasons made current line based SLAM approaches rely on range cameras or laser scanners [162–165].

We propose to tackle all these issues using a purely visual-based approach. Building upon the ORB-SLAM [1] framework, we propose PL-SLAM (Point and Line SLAM), a solution that can simultaneously leverage points and lines information. As recently suggested by [166], lines are parameterized by their endpoints, whose exact location in the image plane is estimated following a two-step optimization process. This representation, besides yielding robustness to occlusions and mis-detections, allows integrating the line representation within the SLAM machinery as if they were points and hence re-use most of the ORB-SLAM [1] architecture. The resulting approach is shown to be very accurate in poorly textured environments, and also, improves the performance of the original ORB-SLAM [1] in highly textured sequences (see Fig. 3.5).

We also introduce a new initialization approach that allows estimating an approximate initial map from only line correspondences between three consecutive images. Previous solutions were based on homography [167] or essential matrix estimation [168], and required point

correspondences. To the best of our knowledge, there are no equivalent techniques based on lines. The solution we propose holds on the assumption of constant rotation between three consecutive frames and that these rotations are relatively small. In the experimental subsection, we will show that despite these approximations, the initial map we estimate highly resembles those obtained by point-based solutions, and therefore, are a very good alternative to use when feature points are not available.

In summary, this section main contributions are: 1) an algorithm for scene 3D reconstruction and camera pose estimation robust to poorly textured scenarios that builds upon the joint estimation of point and line correspondences; and 2) the first—to the best of our knowledge—a new initialization approach that allows estimating an approximate initial map from only line correspondences between three consecutive images.

3.3.1 PL-SLAM: Real-Time Monocular Visual SLAM with Points and Lines

The pipeline of our approach highly resembles that of the ORB-SLAM [1], in which we have integrated the information provided by line features (see Fig. 3.1). We next briefly review the main building blocks in which line operations are performed. For a description of the operations involving point features, the reader is referred to [1].

One of the main issues to address in SLAM algorithms is the computational complexity. In order to preserve the real-time characteristics of ORB-SLAM [1], we have carefully chosen, used and implemented fast methods for operating with lines in all stages of the pipeline: detection, triangulation, matching, culling, relocalization and optimization. Line segments in an input frame are detected by mean of LSD [169], an $O(n)$ line segment detector, where n is the number of pixels in the image. Then, lines are pairwise matched with lines already present in the map using a relational graph strategy [170]. This approach relies on lines' local appearance (Line Band Descriptors) and geometric constraints and is shown to be quite robust against image artifacts while preserving the computational efficiency.

As it is done with point features, after having obtained an initial set of map-to-image line feature pairs, all lines of the local map are projected onto the image to find further correspondences. Then, if the image contains sufficient new information about the environment, it is flagged as a keyframe and its corresponding lines are triangulated and added to the map. To discard possible outliers, lines seen from less than three viewpoints or in less than 25% of the frames from which they were expected to be seen are also discarded (culling). Line positions in the map are optimized with a local Bundle Adjustment (BA). Note in Fig. 3.1 that we do not use lines for loop closing. Matching lines across the whole map is too computationally expensive.

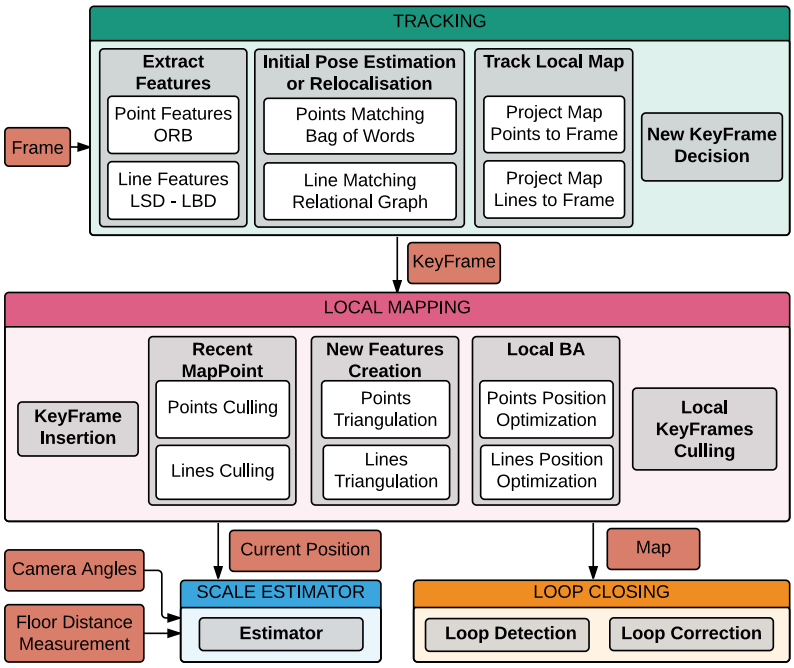


Figure 3.1: **PL-SLAM overview**. The system is an extension of ORB-SLAM [1], and it is composed by four main threads: *Tracking*, *Local Mapping*, *Loop Closing* and *Scale Estimator*. The *Tracking* thread estimates the camera position and decides when to add new keyframes. Then, *Local Mapping* adds the new keyframe information into the map and optimizes it with Bundle Adjustment (BA). The *Loop Closing* thread is constantly checking for loops and correcting them. The *Scale Estimator* estimates the real scale factor by comparing the real robot height vs the estimated one.

Hence, only point features are used for loop detection.

Another issue of monocular SLAM is the lack of scale. In this section we also extend the system with a scale estimator with the use of a cheap pointer sensor mounted on the camera rig.

3.3.2 Line-based SLAM

We next describe the line parameterization and error function we use and how this is integrated within the main building blocks of the SLAM pipeline, namely BA, global relocalization and feature matching.

Line-based Reprojection Error

In order to extend the ORB-SLAM [1] to lines, we need a proper definition of the reprojection error and line parameterization.

Following [166], let $\mathbf{P}, \mathbf{Q} \in \mathbb{R}^3$ be the 3D endpoints of a line, $\mathbf{p}_d, \mathbf{q}_d \in \mathbb{R}^2$ their 2D detections

in the image plane, and $\mathbf{p}_{\mathrm{d}}^{\mathrm{h}}, \mathbf{q}_{\mathrm{d}}^{\mathrm{h}} \in \mathbb{R}^3$ theirs corresponding homogeneous coordinates. From the latter we can obtain the normalized line coefficients as:

$$\mathbf{l} = \frac{\mathbf{p}_{\mathrm{d}}^{\mathrm{h}} \times \mathbf{q}_{\mathrm{d}}^{\mathrm{h}}}{\left|\mathbf{p}_{\mathrm{d}}^{\mathrm{h}} \times \mathbf{q}_{\mathrm{d}}^{\mathrm{h}}\right|}. \tag{3.1}$$

The *line reprojection error* $\mathrm{E}_{\mathrm{line}}$ is then defined as the sum of point-to-line distances E_{pl} between the projected line segment endpoints, and the detected line in the image plane (see Fig. 3.2-right). That is:

$$\mathrm{E}_{\mathrm{line}}(\mathbf{P}, \mathbf{Q}, \mathbf{l}, \boldsymbol{\theta}, \mathbf{K}) = \mathrm{E}_{\mathrm{pl}}^2(\mathbf{P}, \mathbf{l}, \boldsymbol{\theta}, \mathbf{K}) + \mathrm{E}_{\mathrm{pl}}^2(\mathbf{Q}, \mathbf{l}, \boldsymbol{\theta}, \mathbf{K}), \tag{3.2}$$

with:

$$\mathrm{E}_{\mathrm{pl}}(\mathbf{P}, \mathbf{l}, \boldsymbol{\theta}, \mathbf{K}) = \mathbf{l}^\top \pi(\mathbf{P}, \boldsymbol{\theta}, \mathbf{K}), \tag{3.3}$$

where $\mathbf{l}$ are the detected line coefficients, $\pi(\mathbf{P}, \boldsymbol{\theta}, \mathbf{K})$ represents the projection of the endpoint $\mathbf{P}$ onto the image plane, given the internal camera calibration matrix $\mathbf{K}$, and the camera parameters $\boldsymbol{\theta} = \{\mathbf{R}, \mathbf{t}\}$ that include the rotation and translation.

Note that in practice, due to real conditions such as line occlusions or mis-detections, the image detected endpoints $\mathbf{p}_{\mathrm{d}}$ and $\mathbf{q}_{\mathrm{d}}$ will not match the projections of the endpoints $\mathbf{P}$ and $\mathbf{Q}$ (see Fig. 3.2-left). Therefore, we define the detected line reprojection error as:

$$\mathrm{E}_{\mathrm{line,d}}(\mathbf{p}_{\mathrm{d}}, \mathbf{q}_{\mathrm{d}}, \mathbf{l}) = \mathrm{E}_{\mathrm{pl,d}}^2(\mathbf{p}_{\mathrm{d}}, \mathbf{l}) + \mathrm{E}_{\mathrm{pl,d}}^2(\mathbf{q}_{\mathrm{d}}, \mathbf{l}), \tag{3.4}$$

where $\mathbf{l}$ are the projected 3D line coefficients and the detected point-to-line error is $\mathrm{E}_{\mathrm{pl,d}}(\mathbf{p}_{\mathrm{d}}, \mathbf{l}) = \mathbf{l}^\top \mathbf{p}_{\mathrm{d}}$.

Based on the methodology proposed in [166], a recursion over the detected reprojection line error will be applied in order to optimize the pose parameters $\boldsymbol{\theta}$ while approximating $\mathrm{E}_{\mathrm{line,d}}$ to the line error $\mathrm{E}_{\mathrm{line}}$ defined on Eq. (3.2).

Bundle Adjustment with Points and Lines

The camera pose parameters $\boldsymbol{\theta} = \{\mathbf{R}, \mathbf{t}\}$ are optimized at each frame with a BA strategy that constrains $\boldsymbol{\theta}$ to lie in the SE(3) group. For doing this, we build upon the framework of the ORB-SLAM [1] but besides feature point observations, we include the lines as defined in the previous subsection. We next define the specific cost function we propose to be optimized by the BA that combines the two types of geometric entities.

Let $\mathbf{X}_j \in \mathbb{R}^3$ be the generic j-th point of the map. For the i-th keyframe, this point can be

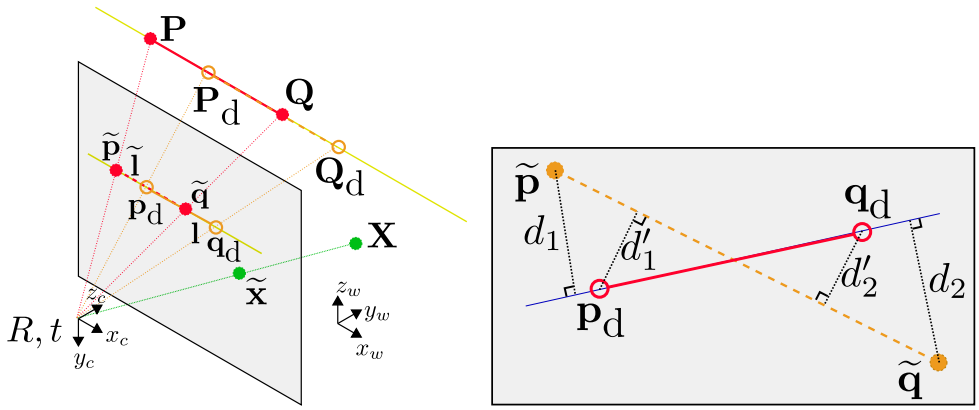


Figure 3.2: **Points and lines notation. Left**: Notation. Let $\mathbf{P}, \mathbf{Q} \in \mathbb{R}^3$ be the 3D endpoints of a 3D line, $\tilde{\mathbf{p}}, \tilde{\mathbf{q}} \in \mathbb{R}^2$ their projected 2D endpoints to the image plane and $\tilde{\mathbf{l}}$ the projected line coefficients. $\mathbf{p}_\mathrm{d}, \mathbf{q}_\mathrm{d} \in \mathbb{R}^2$ the 2D endpoints of a detected line, $\mathbf{P}_\mathrm{d}, \mathbf{Q}_\mathrm{d} \in \mathbb{R}^3$ their real 3D endpoints, and $\mathbf{l}$ the detected line coefficients. $\mathbf{X} \in \mathbb{R}^3$ is a 3D point and $\tilde{\mathbf{x}} \in \mathbb{R}^2$ its corresponding 2D projection. **Right**: Line-based reprojection error. d_1 and d_2 represent the *line reprojection error*, and d'_1 and d'_2 the *detected line reprojection error* between a detected 2D line (blue solid) and the corresponding projected 3D line (green dashed).

projected onto the image plane as:

$$\tilde{\mathbf{x}}_{i,j} = \pi(\mathbf{X}_j, \boldsymbol{\theta}_i, \mathbf{K}), \tag{3.5}$$

where $\boldsymbol{\theta}_i = \{\mathbf{R}_i, \mathbf{t}_i\}$ denotes the specific pose of the i-th keyframe. Given an observation $\mathbf{x}_{i,j}$ of this point, we define following 3D error:

$$\mathbf{e}_{i,j} = \mathbf{x}_{i,j} - \tilde{\mathbf{x}}_{i,j} \ . \tag{3.6}$$

Similarly, let us denote by $\mathbf{P}_j$ and $\mathbf{Q}_j$ the endpoints of the j-th map line segment. The corresponding image projections (expressed in homogeneous coordinates) onto the same keyframe can be written as:

$$\tilde{\mathbf{p}}^\mathrm{h}_{i,j} = \pi(\mathbf{P}_j, \boldsymbol{\theta}_i, \mathbf{K}), \tag{3.7}$$

$$\tilde{\mathbf{q}}^\mathrm{h}_{i,j} = \pi(\mathbf{Q}_j, \boldsymbol{\theta}_i, \mathbf{K}) \ . \tag{3.8}$$

Then, given the image observations $\mathbf{p}_{i,j}$ and $\mathbf{q}_{i,j}$ of the j-th line endpoints, we use Eq. (3.1) to estimate the coefficients of the observed line $\tilde{\mathbf{l}}_{i,j}$. We define the following error vectors for the

line:

$$\mathbf{e}'_{i,j} = (\tilde{\mathbf{l}}_{i,j})^\top (\mathbf{K}^{-1}\mathbf{p}^{\mathrm{h}}_{i,j}), \tag{3.9}$$

$$\mathbf{e}''_{i,j} = (\tilde{\mathbf{l}}_{i,j})^\top (\mathbf{K}^{-1}\mathbf{q}^{\mathrm{h}}_{i,j}). \tag{3.10}$$

The errors (3.9, 3.10) are in fact instances of the point-to-line error (3.3). As explained in [166] they are not constant w.r.t. shift of the endpoints $\mathbf{P}_j$, $\mathbf{Q}_j$ along the corresponding 3D line, which serves as implicit regularization allowing us to use such a non-minimal line parametrization in the BA.

Observe that representing lines using their endpoints we obtain comparable error representations for points and lines. We can therefore build a unified cost function that integrates each error term as:

$$C = \sum_{i,j} \rho \left(\mathbf{e}_{i,j}^\top \mathbf{\Omega}_{i,j}^{-1} \mathbf{e}_{i,j} + \mathbf{e}_{i,j}^{'\top} \mathbf{\Omega}_{i,j}^{'-1} \mathbf{e}'_{i,j} + \mathbf{e}_{i,j}^{''\top} \mathbf{\Omega}_{i,j}^{''-1} \mathbf{e}''_{i,j} \right) \tag{3.11}$$

where ρ is the Huber robust cost function and $\mathbf{\Omega}_{i,j}$, $\mathbf{\Omega}'_{i,j}$, $\mathbf{\Omega}''_{i,j}$ are the covariance matrices associated to the scale at which the keypoints and line endpoints were detected, respectively.

Global Relocalization

An important component of any SLAM method, is an approach to relocalize the camera when the tracker is lost. This is typically achieved by means of a Perspective-n-Point (PnP) algorithm, that estimates the pose of the current (lost) frame given correspondences with 3D map points appearing in previous keyframes. On top of the PnP method, a RANSAC strategy is used to reject outliers correspondences.

In the ORB-SLAM [1], the specific PnP method that is used is the EPnP [171], which however, only accepts point correspondences as inputs. In order to make our approach appropriate to handle lines for relocalization, we have replaced the EPnP by the recently published EPnPL [166], which minimizes the *detected line reprojection error* of Eq. (3.4).

Furthermore, EPnPL [166] is robust to partial line occlusion and mis-detections. This is achieved by means of a two-step procedure in which first minimizes the reprojection error of the detected lines and estimates the line endpoints $\mathbf{p}_{\mathrm{d}}, \mathbf{q}_{\mathrm{d}}$. These points, are then shifted along the line in order to match the projections $\tilde{\mathbf{p}}_{\mathrm{d}}, \tilde{\mathbf{q}}_{\mathrm{d}}$ of the 3D model endpoints $\mathbf{P}, \mathbf{Q}$ (see Fig. 3.2). Once these matches are established, the camera pose can be reliably estimated.

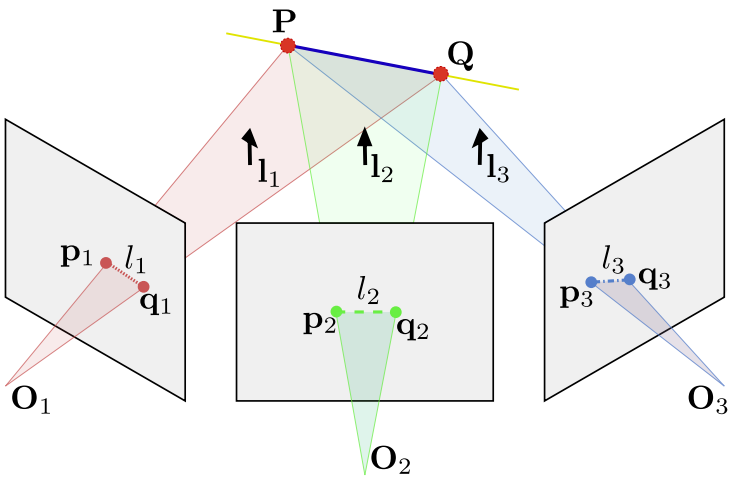


Figure 3.3: **Camera rotation estimation from line correspondences.** $\mathbf{P}, \mathbf{Q} \in \mathbb{R}^3$ are the 3D line endpoints, l_i, $i = \{1, 2, 3\}$ their detections in three consecutive frames with endpoints $\mathbf{p}_i, \mathbf{q}_i$, and coefficients $\mathbf{l}_i$.

Map Initialization with Lines

Another contribution of this section is an algorithm to estimate an initial map using only line correspondences. Current optimization-based SLAM approaches are initialized with maps built from point correspondences between at least two frames. Homography [167] or essential matrix [168] estimation algorithms are then used to compute the initial map and pose parameters. We next describe our line-based solution for map initialization, which can be a good alternative in low textured scenes with lack of feature points.

Let us consider the setup of Fig. 3.3, where a line defined by endpoints $\mathbf{P}, \mathbf{Q}$ is projected onto three camera views. Let $\{\mathbf{p}_1, \mathbf{q}_1\}$, $\{\mathbf{p}_2, \mathbf{q}_2\}$ and $\{\mathbf{p}_3, \mathbf{q}_3\}$ be the endpoint projections in each of the views and $\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}_3 \in \mathbb{R}^3$ the corresponding line coefficients computed from the projected endpoints.

We will make the assumption of small and continuous rotation between consecutive camera poses, such that the rotation from the first to the second camera views is the same than the rotation from the second to the third one[1]. Under this assumption we can represent the three camera rotations by $\mathbf{R}_1 = \mathbf{R}^\top$, $\mathbf{R}_2 = \mathbf{I}$, and $\mathbf{R}_3 = \mathbf{R}$, with $\boldsymbol{I}$ being the 3×3 identity matrix.

Note that the line coefficients $\mathbf{l}_i$, $i = \{1, 2, 3\}$ also represent the parameters of a vector which is normal to the plane formed by the center of projection $\mathbf{O}_i$ and the projections $\mathbf{p}_i, \mathbf{q}_i$. The cross product of two such vectors $\mathbf{l}_i$ will be parallel to the line $\mathbf{P}, \mathbf{Q}$ and at the same time orthogonal to the third vector, all of them appropriately rotated and put in a common reference. This

[1]In the experimental subsection we evaluate the consequences of this assumption, and show that in practice is a good approximation.

constraint can be written as:

$$\mathbf{l}_2^\top \left((\mathbf{R}^\top \mathbf{l}_1) \times (\mathbf{R}\mathbf{l}_3) \right) = 0. \tag{3.12}$$

Additionally, for small rotations we can approximate $\mathbf{R}$ as:

$$\mathbf{R} = \begin{pmatrix} 1 & -r_3 & r_2 \\ r_3 & 1 & -r_1 \\ -r_2 & r_1 & 1 \end{pmatrix}. \tag{3.13}$$

For this parametrization, having three matched lines, we will have three quadratic equations like Eq. (3.12) with three unknowns, r_1, r_2 and r_3. We adapt the polynomial solver of [172], which yields up to eight solutions. For each possible rotation matrix we can get $\mathbf{t}_1, \mathbf{t}_3$ by using the trifocal tensor equations [173] which will be linear in $\mathbf{t}_1, \mathbf{t}_3$. We assume $\mathbf{t}_2 = \mathbf{0}$. We evaluate the eight possible solutions and keep the one that minimizes Eq. (3.12).

It is worth to point that in order to get enough independent constraints when solving for the translation components using the trifocal tensor equations, we need two additional line correspondences, and hence, the total number of line matches required by our algorithm is five.

3.3.3 Scale Estimation

It is well-known that monocular SLAM methods are up to scale, but for real tasks it is necessary to have an approximate real scale estimate of the map, trajectory and actual position of the robot. To this extend, we have further improved the system by adding a scale estimation module.

The scale estimator thread computes the scale as the factor between the real height of the robot/camera and the estimated one. The real height is obtained by correcting the distance measurement of a lidar pointer pointing to the ground with the camera gyroscope as:

$$h' = h \cos\alpha \cos\sigma, \tag{3.14}$$

where h' is the real height, h the lidar pointer range measurement, α the roll angle and σ the pitch angle. Notice that this algorithm adds the constraint that the floor under the robot must be flat in the first frames until the scale estimator has converged.

3.3.4 Experimental Evaluation

We have compared our system with the current state-of-the-art Visual SLAM methods using the TUM RGB-D benchmark [13]. Also, we evaluate the proposed initialization approach with synthetic and real data and compare the computation time of our PL-SLAM algorithm and the

TUM RGB-D Sequence	PL-SLAM	PL-SLAM Classic Init	LAP-SLAM	ORB-SLAM	PTAM†	LSD-SLAM†	RGBD-SLAM†
f1_xyz	1.21	1.46	1.34	1.38	**1.15**	9.00	1.34
f2_xyz	0.43	1.49	0.46	0.54	**0.2**	2.15	2.61
f1_floor	7.59	9.42	7.85	8.71	-	38.07	**3.51**
f2_360_kidnap	3.92	60.11	4.12	4.99	**2.63**	-	393.3
f3_long_office	**1.97**	5.33	2.45	4.05	-	38.53	-
f3_nstr_tex_far	ambiguity detected	37.60	-	ambiguity detected	34.74	**18.31**	-
f3_nstr_tex_near	2.06	**1.58**	1.31	2.88	2.74	7.54	-
f3_str_tex_far	**0.89**	1.25	-	0.98	0.93	7.95	-
f3_str_tex_near	1.25	7.47	1.31	1.5451	**1.04**	-	-
f2_desk_person	**1.99**	6.34	2.75	5.95	-	31.73	6.97
f3_sit_xyz	**0.066**	9.03	0.27	0.08	0.83	7.73	-
f3_sit_halfsph	**1.31**	9.05	1.76	1.48	-	5.87	-
f3_walk_xyz	**1.54**	ambiguity detected	1.54	1.64	-	12.44	-
f3_walk_halfsph	**1.60**	ambiguity detected	1.76	2.09	-	-	-

Table 3.1: **Localization accuracy in the TUM RGB-D Benchmark**: Euclidean distance median in centimeters over 5 executions for each sequence. All trajectories were aligned with 7DoF with the ground truth before computing the Absolute Trajectory Error (ATE) error with the script provided by the benchmark [13]. Both ORB-SLAM and PL-SLAM were executed with the parametrization of the on-line open source ORB-SLAM package. †Results of PTAM, LSD-SLAM and RGBD-SLAM were extracted from [1].

ORB-SLAM [1]. All experiments were carried out with an Intel Core i7-4790 (4 cores @3.6 GHz), 8Gb Random-Access Memory (RAM). Due to the randomness of some stages of the pipeline, e.g., initialization, position optimization or global relocalization, all experiments were run five times and we report the median of all executions.

Localization Accuracy in the TUM RGB-D Benchmark

To evaluate the localization accuracy we compare our PL-SLAM method against current state-of-the-art Visual SLAM methods, including ORB-SLAM [1], PTAM [54], LSD-SLAM [60] and RGBD-SLAM [161]. The metric used for the comparison is the Absolute Trajectory Error (ATE), provided by the evaluation script of the benchmark. Before computing the error, all trajectories are aligned using a similarity warp except for the RGBD-SLAM [161] which is aligned by a rigid body transformation. The results are summarized in Table 3.1.

Note that our PL-SLAM consistently improves the trajectory accuracy of ORB-SLAM [1] in

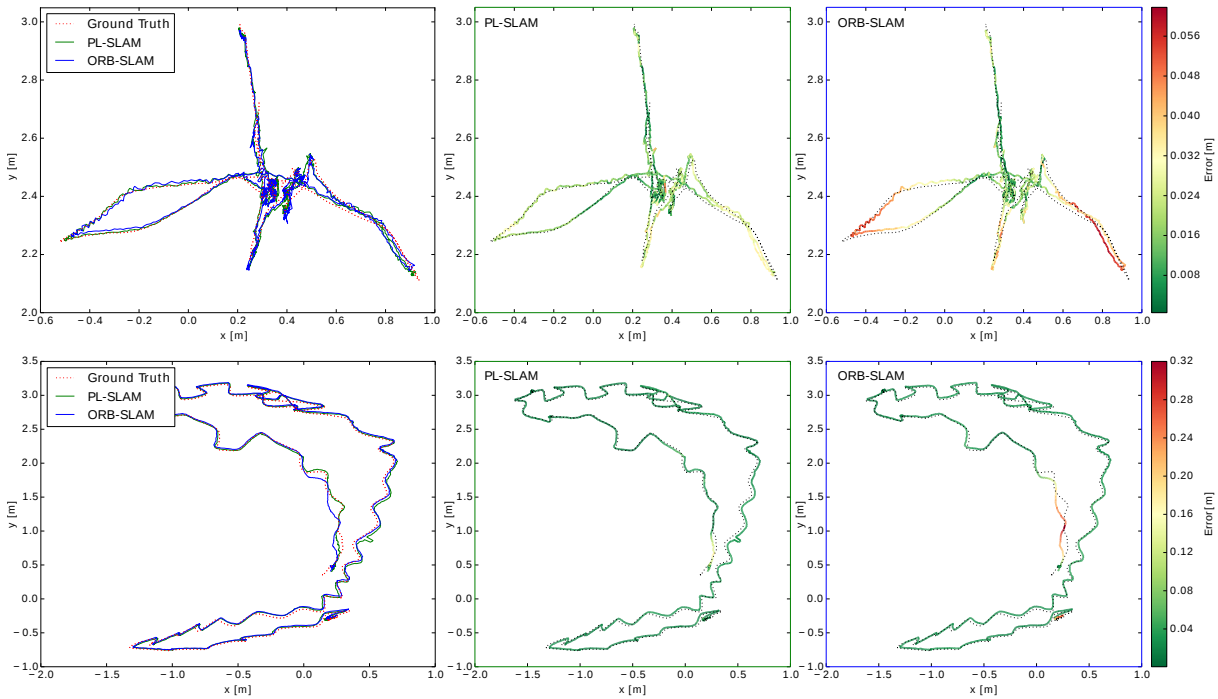


Figure 3.4: **ORB-SLAM vs. PL-SLAM.** Comparison between ORB-SLAM and PL-SLAM in two TUM-RGBD sequences. **First-Column**: Ground truth and predicted trajectories by PL-SLAM and ORB-SLAM. The dotted line represents the ground truth, the green line the trajectory obtained with PL-SLAM, and the blue line the trajectory obtained with ORB-SLAM. **Second-Column and Third-Column**: Estimated trajectories color-coded with the amount of error by PL-SLAM and ORB-SLAM, respectively.

all sequences. Indeed, it yields the best result in all but two sequences, for which PTAM [54] performs slightly better. Nevertheless, PTAM [54] turned not to be so reliable, as in 5 out of all 12 sequences it lost track. LSD-SLAM [60] and RGBD-SLAM [161] also lost track in 3 and 7 sequences, respectively.

Fig. 3.4 presents a comparison between ORB-SLAM and our extension PL-SLAM in sequences *freiburg2_desk_person* and *freiburg3_sit_halfsph*. Note how PL-SLAM reduces the drift w.r.t ORB-SLAM by adding line geometric primitives information.

Localization Accuracy in Low Textured Scenarios

To demonstrate the performance of PL-SLAM in low textured scenarios we performed real experiments in two environments, each with a predominant non-textured element: planes or cylinders.

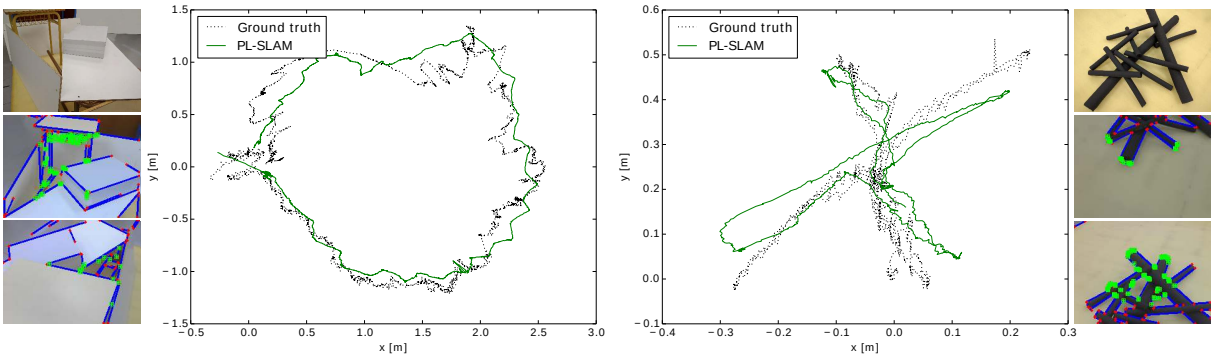


Figure 3.5: **Results on non-textured scenes. Left**: Localization accuracy in structured low texture scenarios. **Right**: Low texture sequence. For each block: **Top-Image**: Scene overview. **Middle and Bottom Images**: Sampled frames from the video sequence. **Plot**: Ground truth (green dotted line) vs. predicted trajectory (green solid line).

The video data was recorded at 30 Hz frame rate and a sensor resolution of 640x480 pixels. The ground-truth trajectory was obtained from a rigid body motion-capture system with 13 high-speed tracking cameras at 120 FPS.

In the first experiment we evaluated our PL-SLAM method in a scene with predominant non-textured planes (Fig. 3.5-left). Despite the absence of texture, line features are still consistent enough to initialize the map and obtain a trajectory with an Absolute Trajectory Error (ATE) of 0.31 meters.

We also tested our system in a scene with only non-textured cylinder elements (Fig. 3.5-right). This is a challenging scenario because of the predominance of apparent lines - most line features are not lines, but instead apparent lines product of projecting the cylinders to the camera plane. However, the combination of point and line features makes the system still reliable enough to initialize a map and estimate a trajectory with 0.1 meters of ATE.

In both experiments ORB-SLAM fails to estimate a trajectory.

Low Texture Reflectant Glazed Building Experiment

We have also performed a qualitative evaluation of our method tracking the frontal view of a glazed building. This is a challenging task due to the glaze low texture profile and the window reflections. Fig. 3.6 shows the results of the experiment. The system is robust enough to filter the features of the reflected image by checking movement consistency between frames and relay on line features of the building window's frames to estimate a trajectory.

As the experiment was performed in a non-controlled environment, we do not have the ground truth trajectory, and we only provide an estimate one.

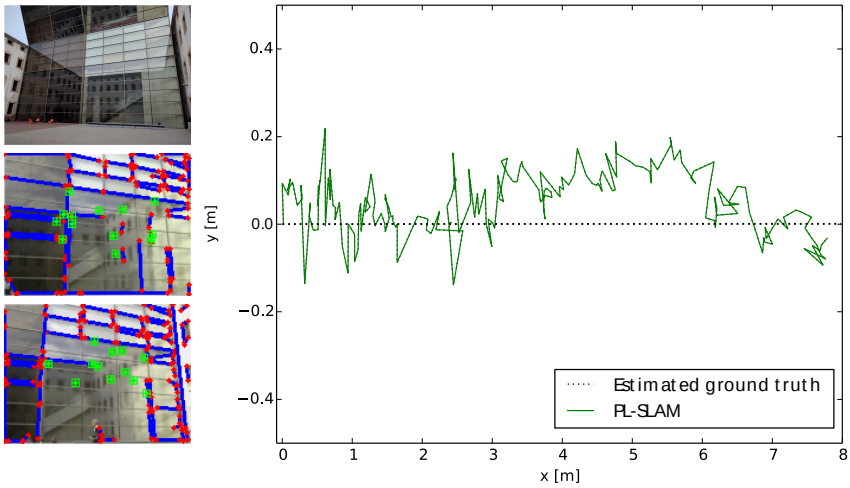


Figure 3.6: **Localization of a glazed building. Top**: Scene overview. **Middle and Bottom**: Sampled frames from the video sequence. **Center**: Approximated ground truth (green dotted line) vs. predicted trajectory (green solid line).

Real Robot Qualitative Experiment

To further test the proposed system, we have embedded PL-SLAM in a quadcopter framed robot. For the real robot experimentation we have further extended the system by adding the real scale estimator described in Subsec. 3.3.3. The experiment setup was designed to test the most common and challenging situations in real scenarios: high speed direction changes, apparent lines, challenging light conditions, long distance features and inconsistent camera frame rates with low latency peaks. PL-SLAM was capable of running onboard in real-time at 20Hz with a camera resolution of 640x480 pixels on an embeded Intel® NUC Skull. The scale estimate error factor was 0.06. The generated map and estimated trajectory outputs are shown in Fig. 3.7. Because the experiment was performed in a non-controlled environment, we do not have the trajectory ground truth. The scale estimate error was computed as: $\left|1 - \frac{\tilde{s}}{s}\right|$ being s the real factor and $\tilde{s}$ the estimated factor.

Map Initialization - Synthetic Experiments

In order to evaluate the map initialization algorithm we performed several synthetic and real experiments.

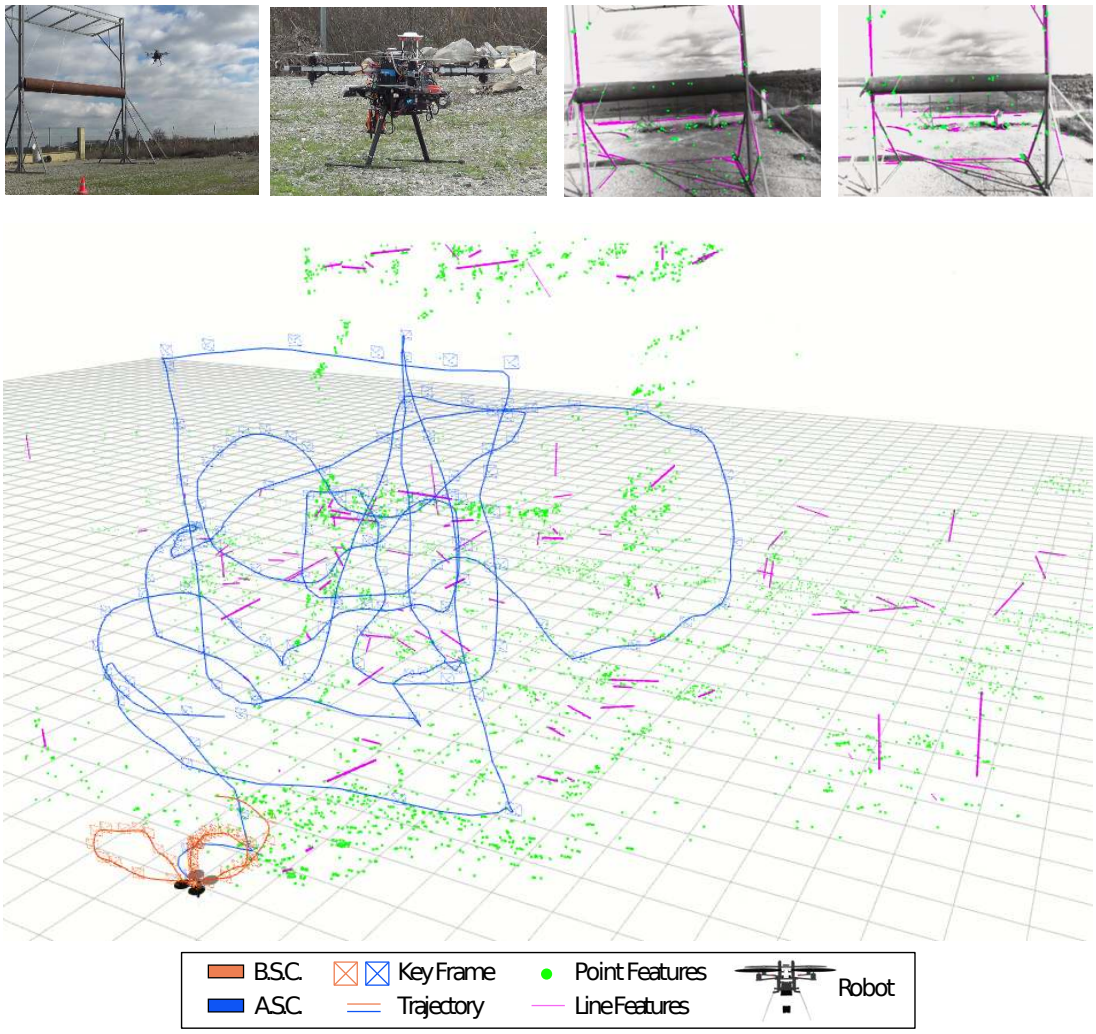


Figure 3.7: **Real robot experiment**. **Top-First**: Scene overview. **Top-Second**: Robot. **Top-Third and Top-Fourth**: Sampled frames from the video sequence. **Bottom**: Experiment visualization of the generated map and estimated trajectory. For both key-frames (crossed boxes) and trajectories (solid lines) orange color stands for *before scale convergence estimation* (B.S.C.) and blue for *after scale convergence estimation* (A.S.C.). Point features are displayed as green dots and line features as pink segments.

In the synthetic tests we first evaluated the stability of the polynomial solver we built, modifying the toolbox of Kukelova *et al.* [172]. Fig. 3.8-left shows the distribution of errors in the parameter estimation for ideal solutions. Note that the average error is around 1e-15, indicating that our modified solver is very stable.

Additionally, we have assessed the consequences of assuming small and constant rotations between three consecutive frames. Fig. 3.8-right displays the rotation and translation errors produced for increasing inter-frame rotations. While the estimated rotation error remains within

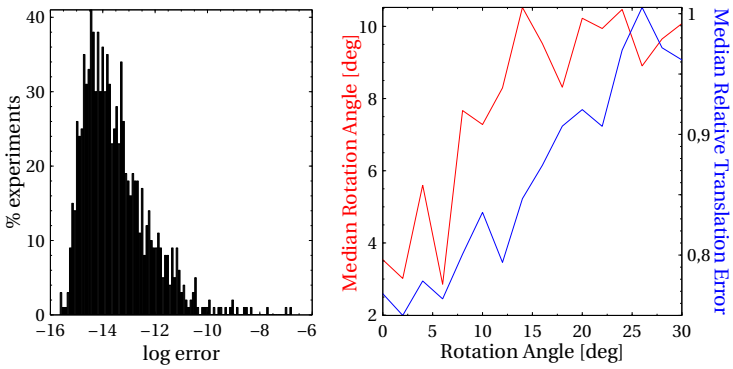


Figure 3.8: **Map initialization - synthetic experiments. Left**: Numerical stability of the polynomial system solver. **Right**: Rotation and translation error w.r.t frames rotation.

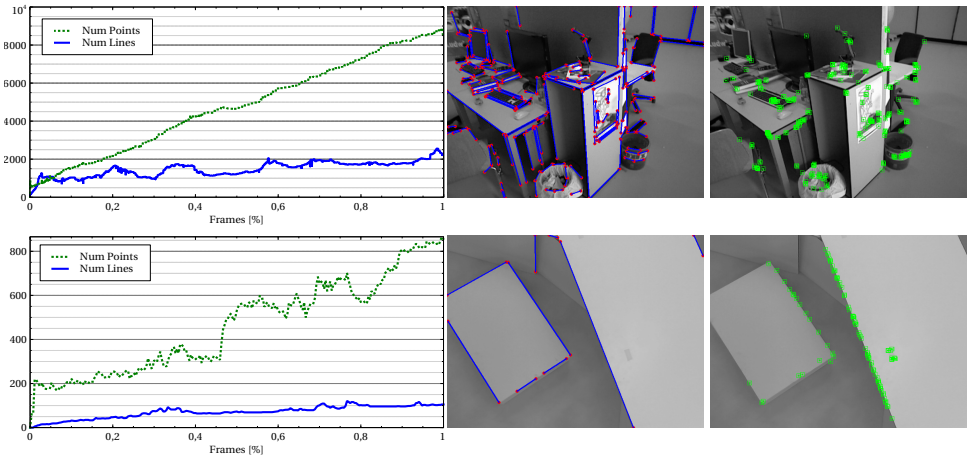


Figure 3.9: **Number of points vs. number of lines.** In high texture (top-row) and low texture (bottom-row). **Left-Column**: Evolution of the number of points and lines in the map. The green dotted line shows the number of points, the blue solid line the number of lines. **Middle-Column and Right-Column**: Visualization of the line and point features of a sample frame, respectively.

relatively small bounds, the translation error is more severely affected by the small rotation assumption. In any event, when this initial map is fed into the BA optimizer, the translation error is drastically reduced.

Map Initialization - Real Experiments

We also evaluated our PL-SLAM method using the classic initialization (based on homography or essential matrix computation), and with the proposed map initialization based only on lines (see again Table 3.1). As expected, the accuracy with the line map initialization drops due to the small rotation assumptions it does. However, in the low textured sequence *f3_nstr_tex_far*, the classic initialization detects an ambiguity which disables it of initializing the map. In contrast,

Thread	Operation	PL-SALM	ORB -SLAM
Local Mapping	KeyFrame Insertion	17.08	9.86
	Map Feature Culling	1.18	1
	Map Features Creation	74.64	8.39
	Local BA	218.25	118.5
	KeyFrame Culling	12.7	2.86
	Total	3Hz	7Hz
Tracking	Features Extraction	31.32	10.76
	Initial Pose Estimation	7.16	7.16
	Track Local Map	12.58	3.18
	Total	20Hz	50Hz

Table 3.2: **Tracking and mapping execution time** Mean execution time in milliseconds of 5 different sequences of the TUM RGB-D benchmark [13].

the proposed line initialization is able to estimate an initial map. In the sequences *f3_walk_xyz* and *f3_walk_halfsph* the proposed initialization does not work due lo large inter-frame rotations produced in the initial frames.

Number of Points versus Number of Lines

We finally perform two experiments to analyze the behaviour of point and line features in low and high textured scenarios (see Fig. 3.9). In the first experiment both features are examined in a high texture scenario shown in Fig. 3.9-top. As expected, point features predominate over line features. It may seem like lines are dispensable due to the low number of constraints that are adding to the BA. However, notice that lines contain more information (infinit points) and are more robust than a single point causing a consistent improvement in the trajectory estimation even in high textured environments as shown in Table 3.1.

In the second experiment, the system is tested in. The challenging low textured environment shown in Fig. 3.9-bottom. In this case, the number of points is still larger than the number of lines because several key-points lie on each line. However, while lines are consistent across frames, points are miss-paired due to the similarity of its descriptor.

Computation Time

While adding line primitives to the visual SLAM improves accuracy and robustness, it also increases the computational complexity. Table 3.2 summarizes the time required for each subtask within the *Tracking* and *Local Mapping* blocks, for PL-SLAM and ORB-SLAM [1]. Note that the subtasks with larger penalties are the map features creation and the local BA. In any event the final frame rate of the PL-SLAM is near real time (20 FPS) in a standard and not

(a) Reference Mask (b) $\mathcal{L}_{\text{CE}}$ (c) $\mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{S}}$ (d) $\mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{T}}$ (e) $\mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{S}} + \mathcal{L}_{\text{T}}$

Figure 3.10: **FaSTGAN overview.** The proposed approach learns spatio-temporal object models given a reference mask. When only training with cross-entropy ($\mathcal{L}_{\text{CE}}$) large errors at the object boundary are produced (b). Adding spatial consistency ($\mathcal{L}_{\text{S}}$) improves the latter but errors still occur on semi-occluded parts (c). Instead, using temporal consistency ($\mathcal{L}_{\text{T}}$) improves the mask propagation but the model struggles to recover from occluded parts (d). In this section we leverage the benefit of combining spatio-temporal information (e) running at 32 FPS.

optimized point cloud.

3.4 Object Segmentation and Tracking

In order to reconstruct 3D objects we first need to detect and track them. With this objetive in mind, in this section, we tackle the problem of semi-supervised video object segmentation. This consists in segmenting an object from the background throughout a video sequence given its ground truth mask in the initial frame. Large video datasets like DAVIS [67, 68] and the recently released YouTube-VOS [69] have spurred a number of deep networks methods [69, 74–77, 79–81, 83–85, 88–94, 174] that improve by a large margin the performance of approaches from the pre-DL era [70–73, 175]. The problem, however, is still far from being solved. Occlusions, rapid object movements, appearance changes and similarity of different instances of the same object are still a major obstacle that often require heavy post-processing operations, human intervention and expensive model fine-tuning.

In order to achieve robustness to these challenges, descriptive spatio-temporal object models encoding appearance and geometric changes need to be learned. Most existing state-of-the-art approaches [74, 75, 80] rely on a reference mask to fine-tune the model and in some cases, use the previous mask as a guidance. Formally, if we denote this reference mask by $\mathbf{Y}_0$ and the RGB frame at time t by $\mathbf{X}_t$, these approaches model the mask $\hat{\mathbf{Y}}_t$ as $p(\hat{\mathbf{Y}}_t|\mathbf{Y}_0, \mathbf{X}_0, \mathbf{X}_{t-1}, \mathbf{X}_t)$ or $p(\hat{\mathbf{Y}}_t|\mathbf{Y}_0, \mathbf{X}_0, \mathbf{X}_{t-1}, \mathbf{X}_t, \hat{\mathbf{Y}}_{t-1})$. Since no temporal consistency is enforced, these methods tend to be robust to drifting, but they underperform when the object drastically changes its appearance.

This can be remedied by leveraging the temporal consistency of the segmented mask. However, while this was a common practice in the past [71–73], it is not usual among Deep Learning methods, in part due to the absence of large scale video object segmentation datasets. Very recently, Xu *et al.* [69] used a convolutional LSTM trained with the YouTube-VOS dataset to learn

long-term temporal dependencies from the entire history of the object in the video. That is, the mask $\hat{\mathbf{Y}}_t$ is modeled as $p(\hat{\mathbf{Y}}_t|\mathbf{Y}_0, \mathbf{X}_0, \mathbf{X}_1, \ldots, \mathbf{X}_t)$. While this approach demonstrates improved performance compared to previous baselines which did not enforce temporal consistency, it seems to be too generic, as it still needs a computationally demanding fine-tuning step when applied to a sequence with unseen objects.

In this section, we propose FaSTGAN (see Fig. 3.10), an intermediate solution that learns spatio-temporal object appearance models over finite time horizons that does not require fine-tuning nor post-processing. Essentially, during training, we model the segmentation masks as $p(\hat{\mathbf{Y}}_t|\mathbf{Y}_0, \mathbf{X}_0, \mathbf{X}_{t-K}, \ldots, \mathbf{X}_t)$, where K is the size of the temporal window. In order to implement this model, we design a regressor network architecture inspired by the agile siamese encoder-decoder structure proposed by Wug *et al.* [81]. In its original form, this regressor is only fed by the reference mask and the masked image at the previous time step. To exploit all information within a temporal window of size K, we could naively make the regressor have access to more information by concatenating features from the K previous frames. This, however, would heavily penalize the efficiency and adaptability of the model. We have therefore devised a novel GAN architecture (Fig. 3.11) in which, during training, this regressor is combined with $K + 1$ discriminators that enforce the temporal and spatial coherence of the generated masks over the temporal window. At test, these discriminators are removed, keeping the original efficiency of the siamese regressor, while allowing it to model the object across longer time horizons.

As a result, our architecture only uses video data to train and does not require any kind of fine-tuning nor post-processing operations at test time. This makes our approach very efficient, running at 32 FPS on 512×512 video frames, being about $4\times$ faster than [81], which was the fastest video segmentation method so far with 7.7 FPS reported in their original work. Furthermore, we demonstrate the accuracy of our method to be on par with state-of-the-art techniques on single-object video segmentation.

In summary, this section main contributions are: 1) a method to learn spatio-temporal object models for real-time detection and tracking from a single reference image; and 2) a novel generative architecture that enforce spatio-temporal consistency over finite temporal windows that does not require fine-tuning nor post-processing operations when applied to new sequences with unseen objects.

3.4.1 Problem Formulation

We next formally describe our problem, and generalize the formulation introduced in Sec. 3.4 to an arbitrary number of objects, *i.e.*, we aim to design a Deep Learning model able to segment

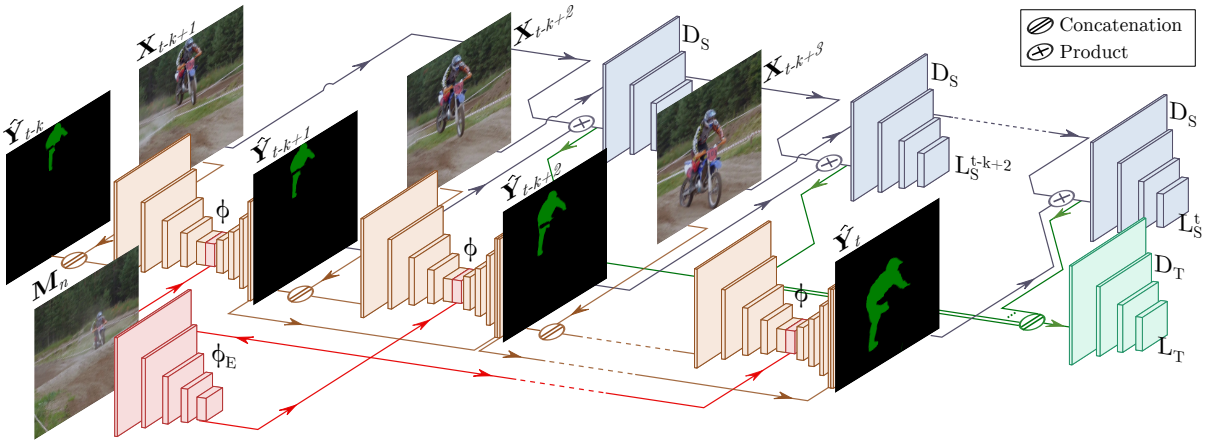


Figure 3.11: **FaSTGAN model.** The diagram displays the model at training time for frames $t-K+1, t-K+2, \ldots, t$. The proposed architecture consists of three main components: a segmentation regressor ϕ, a spatial critic D_{S} and a temporal critic D_{T}. Weights are shared across each network of the same instance. ϕ_{E} is the encoder part of ϕ and it encodes the image-mask pair $\langle \mathbf{X}_t, \hat{\mathbf{Y}}_{t-1} \rangle$ and $\mathbf{M}_n$ at every time step.

and track objects along a video sequence given only one single segmentation mask per object.

Let $\mathbf{x} = (\mathbf{X}_1, \ldots, \mathbf{X}_T)$ be an input RGB video with T frames, where $\mathbf{X}_t \in \mathbb{R}^{H \times W \times 3}$ denotes the t-th frame. Let us also define $\mathbf{m} = (\mathbf{M}_1, \ldots, \mathbf{M}_N)$ as a set of reference segmentations of N objects. The reference segmentation $\mathbf{M}_n \in \mathbb{R}^{H \times W \times (3+1)}$ for the n-th object is the concatenation of the first RGB frame in which the object appears with its annotated binary mask. Our goal is to estimate the masks $\hat{\mathbf{y}}$ of all N objects along the entire sequence $\mathbf{x}$, i.e., we want to learn the mapping $\mathcal{M} : (\mathbf{x}, \mathbf{m}) \rightarrow \hat{\mathbf{y}}$, where $\hat{\mathbf{y}} = (\hat{\mathbf{Y}}_1, \ldots, \hat{\mathbf{Y}}_T)$, and $\hat{\mathbf{Y}}_t \in \mathbb{R}^{H \times W \times N}$ contains the N tracked objects masks in the t-th video frame. We define the ground truth masks $\mathbf{y}$ for a certain sequence $\mathbf{x}$ as $\mathbf{y} = (\mathbf{Y}_1, \ldots, \mathbf{Y}_T)$.

3.4.2 Spatio-Temporal GAN

Fig. 3.11 shows an overview of FaSTGAN, our proposed approach for video object segmentation. A regressor Φ is trained on the binary segmentation task of separating the desired object from the background and two WGAN-GP [44] based critics, D_{S} and D_{T}, enforce the model to produce semantically and temporally consistent estimates. To simplify the model, we introduce a Markov assumption defining the conditional distribution $p(\hat{\mathbf{y}}|\mathbf{x}, \mathbf{m})$ to be factorized as:

$$p(\hat{\mathbf{y}}|\mathbf{x}, \mathbf{m}) = \prod_{t=1}^{T} p(\hat{\mathbf{Y}}_t | \mathbf{x}_{t-K}^{t}, \mathbf{m}, \hat{\mathbf{y}}_{t-K}^{t-1}), \tag{3.15}$$

meaning that we assume objects in a certain frame to be trackable given the reference segmentations $\mathbf{m}$, the current and K-1 previous frames $\mathbf{x}_{t-K}^{t} = (\mathbf{X}_{t-K}, \ldots, \mathbf{X}_t)$, and the K-1 previous estimated segmentation masks $\hat{\mathbf{y}}_{t-K}^{t-1} = (\hat{\mathbf{Y}}_{t-K}, \ldots, \hat{\mathbf{Y}}_{t-1})$.

During training, the regressor Φ is required to learn the mapping $\mathcal{M}$ by modeling the distribution $p(\hat{\mathbf{Y}}_t | \mathbf{x}_{t-K}^{t}, \mathbf{m}, \hat{\mathbf{y}}_{t-K}^{t-1})$, as $\hat{\mathbf{Y}}_t = \Phi(\mathbf{X}_t, \mathbf{m}, \hat{\mathbf{Y}}_{t-1})$. A key property of our design is that the regressor does not directly receive the full information of the K-temporal window. Instead, it is trained to binary segment the K frames, one at a time, given a single foreground/background segmentation $\mathbf{M}_n$ of the desired object $(\mathbf{x}_{t-K}^{t}, \mathbf{M}_n) \rightarrow \hat{\mathbf{y}}_{t-K}^{t}$. Each predicted mask is then independently evaluated by a spatial critic $D_{\mathrm{S}}(\mathbf{X}_i, \hat{\mathbf{Y}}_i)$ $\forall i \in [t-K, t]$ that aims to penalize non-consistent semantic masks per frame. The temporal consistency is assessed by a temporal critic $D_{\mathrm{T}}(\mathbf{x}_{t-K}^{t}, \hat{\mathbf{y}}_{t-K}^{t})$ that jointly evaluates the K segmentation masks. Additionally, to feed the regressor with information from previous estimations we introduce a *temporal skip connection* (see details in the following subsection). Note that with this strategy, the regressor is adapted to produce temporally coherent masks within a horizon of size K, without having to simultaneously process the K frames. This will be crucial to deliver a very fast regressor at test time, when the critics will be discarded. In the following subsections we describe in detail each of these components as well as the proposed training loss:

Segmentation Regressor. Given the current frame $\mathbf{X}_t$, the single-view reference segmentation $\mathbf{M}_n$ of the desired object, and the previous estimate $\hat{\mathbf{Y}}_{t-1}$, the segmentation regressor Φ aims to separate the desired object from the background producing the current estimate mask $\hat{\mathbf{Y}}_t = \Phi(\mathbf{X}_t, \mathbf{M}_n, \hat{\mathbf{Y}}_{t-1})$. We denote the encoder part of Φ as Φ_{E}. Similar to [81], Φ_{E} maps the image-mask pairs $\langle \mathbf{X}_t, \hat{\mathbf{Y}}_{t-1} \rangle \in \mathbb{R}^{H \times W \times (3+1)}$ and $\mathbf{M}_n$ to a shared low-dimensional space. Then, feature matching using global convolutions [176] between both features is performed and fed into the decoder part of Φ to produce the estimated mask $\hat{\mathbf{Y}}_t$. In other words, we train Φ to refine a rough mask from the previous frame $t-1$ to estimate the mask at the current frame t using a reference segmentation of the object $\mathbf{M}_n$.

In order to enforce temporal consistency along time, we extend the architecture from [81] with a “temporal skip connection". To do so, we concatenate features in the last decoder layer of Φ with features extracted by the same layer in the previous frame. To reduce the memory complexity involved, we reduce the number of channels in the previous frame feature map by a factor of $1/8$ with a 3×3 convolution, making the computational cost increase negligible. Adding this connection not only provides the model with information from previous frames but also acts as a simplified model memory state similar to an RNN. Moreover, when training, the gradients of future frame predictions will directly flow into previous estimates guiding the optimization to take into account that an estimated mask at frame t will have a direct impact into future ones.

Spatial Critic. Partial object segmentation masks and background leaks are two of the most frequent errors in segmentation. To avoid them, we introduce a spatial critic network, D_{S}, trained to evaluate the semantic consistency of the pixels in the estimated mask, *i.e.*, we penalize the segmented regions that do not contain one and only one fully covered object ending in its borders without extending into the background. In the experimental subsection, we prove this supervision to be more informative for the model in comparison to just using the cross entropy loss, as it improves its accuracy. The structure of D_{S} resembles that of the PatchGan [52] mapping the product[2] $(\mathbf{X}_t \cdot \hat{\mathbf{Y}}_t) \in \mathbb{R}^{H \times W \times 3}$ to an output matrix $\mathbf{S} \in \mathbb{R}^{H/2^6 \times W/2^6}$ where $\mathbf{S}[i,j]$ is used as a partial function to compute the Earth Mover Distance (EMD) between the distributions of the input overlapping patch ij and the real one. This critic helps to improve the difficult task of defining the mask boundaries while enforcing the model not to produce miss-classified small segmentation blobs around the objects of interest.

Temporal Critic. When tracking an object instance across a video sequence we do not only need to have semantically coherent masks, but these masks must also be consistent across time. To this end, the *temporal critic* D_{T} simultaneously evaluates the current estimate w.r.t. to $K-1$ neighbor frames by learning the mapping $(\mathbf{x}^t_{t-K} \cdot \hat{\mathbf{y}}^t_{t-K}) \rightarrow \mathbf{S}$, where as above $\mathbf{S} \in \mathbb{R}^{H/2^6 \times W/2^6}$ is the overlapping partial scores of a PatchGan based critic. This critic helps to learn relative deformation patterns and plausible absolute motion in the segmentation mask pixel space across time. Also, it enforces the model to generate smooth transitions across mask estimates without large noisy changes.

3.4.3 Learning the Model

The loss function we define contains three terms, namely a *balanced binary cross entropy* loss to penalize pixel-wise masks errors w.r.t. ground-truth annotations; the *spatial consistency loss* to drive the distribution of the estimates to the distribution of the training masks; and the *temporal consistency loss* that penalizes temporally non-consistent masks.

Balanced Binary Cross Entropy Loss. We first define the supervised pixel-wise loss for binary classification. To take into account the imbalance between the number of pixels in the object of interest and the background, we apply the balancing strategy proposed in [177] originally used for contour detection. The balanced binary cross entropy loss $\mathcal{L}_{\text{CE}}$ for K frames is given by:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{K} \sum_{t-K}^{t} \sum_{j \in \mathbf{Y}_t} \Big[\beta \mathbf{Y}_{tj} \log p(\hat{\mathbf{Y}}_{tj} = 1) + (1-\beta)(1-\mathbf{Y}_{tj}) \log p(\hat{\mathbf{Y}}_{tj} = 0) \Big] \tag{3.16}$$

[2] By an abuse of notation we perform the element-wise product on the three RGB channels of $\mathbf{X}_t$.

where $\mathbf{Y}_t$ is the ground truth binary mask and $\beta = |\mathbf{Y}_t^-|/|\mathbf{Y}_t|$ is the percentage of pixels not belonging to the object.

Spatial Consistency Loss. In order to optimize the *spatial critic* D_S parameters and learn the distribution of the training data, we use the modification of the standard GAN min-max strategy game [40] proposed by WGAN-GP [43]. In our initial experiments, we observed that replacing the Jensen-Shannon divergence by the continuous EMD resulted in a more stable training. To introduce the required Lipschitz constraint, we apply the gradient penalty proposed by [44] computed as the norm of the gradients with respect to the critic input. Formally, if we denote the data distribution by P_r, the model distribution by P_g, and the random interpolation distribution between masked images by $P_{\widetilde{\mathbf{x}}}$, the spatial consistency loss $\mathcal{L}_\text{S}$ is given by:

$$\begin{aligned}\mathcal{L}_\text{S} = \frac{1}{K}\sum_{t-K}^{t}\Big[\mathbb{E}_{\mathbf{Y}_t\sim P_r}\left[D_\text{S}(\mathbf{X}_t\cdot\mathbf{Y}_t)\right] - \mathbb{E}_{\hat{\mathbf{Y}}_t\sim P_g}[D_\text{S}(\mathbf{X}_t\cdot\hat{\mathbf{Y}}_t)]\Big] \\ - \frac{1}{K}\sum_{t-K}^{t}\lambda_\text{gp}\mathbb{E}_{\widetilde{\mathbf{x}}\sim P_{\widetilde{\mathbf{x}}}}\left[(\|\nabla_{\widetilde{\mathbf{x}}}D_\text{S}(\widetilde{\mathbf{x}})\|_2 - 1)^2\right],\end{aligned} \tag{3.17}$$

where $\widetilde{\mathbf{x}}$ is the random interpolation between $\langle\mathbf{X}_t\cdot\mathbf{Y}_t, \mathbf{X}_t\cdot\hat{\mathbf{Y}}_t\rangle$ and λ_gp is the penalty coefficient.

Temporal Consistency Loss. With the previously defined losses, the segmentation regressor Φ is enforced to estimate pixel-wise spatial-consistent masks. However, there is no constraint to guarantee temporal consistency, meaning that the predicted masks should cover similar content across frames. With the *temporal critic* D_T, we push Φ to maintain temporal consistency by enforcing similarity between joint distributions of K estimated and annotated masks. To estimate the distance between the distributions, we use the approximated Kantorovich-Rubinstein duality [178] of the EMD as proposed in [44]:

$$\mathcal{L}_\text{T} = \mathbb{E}_{x\sim P_r}\left[D_\text{T}(x)\right] - \mathbb{E}_{\hat{x}\sim P_g}[D_\text{T}(\hat{x}) - \lambda_\text{gp}\mathbb{E}_{\widetilde{\mathbf{x}}\sim P_{\widetilde{\mathbf{x}}}}\left[(\|\nabla_{\widetilde{\mathbf{x}}}D_\text{T}(\widetilde{\mathbf{x}})\|_2 - 1)^2\right], \tag{3.18}$$

where $x = \mathbf{x}_{t-K}^t\cdot\mathbf{y}_{t-K}^t$ and $\hat{x} = \mathbf{x}_{t-K}^t\cdot\hat{\mathbf{y}}_{t-K}^t$ are the real and estimated conditional distributions respectively, and $\widetilde{\mathbf{x}}$ is the random interpolation between $\langle x, \hat{x}\rangle$. Note that, again, by an abuse of notation, we extended the element-wise product between each $\mathbf{x}_t$ and $\mathbf{y}_t$ (or $\hat{\mathbf{y}}_t$) along the 3 color channels of $\mathbf{x}_t$.

Overall Loss. To learn to track an object instance across time, we finally define the following minmax problem:

$$\Phi^\star = \arg\min_{\Phi}\max_{D\in\mathcal{D}}\left(\lambda_\text{CE}\mathcal{L}_\text{CE} + \lambda_\text{S}\mathcal{L}_\text{S} + \lambda_\text{T}\mathcal{L}_\text{T}\right) \tag{3.19}$$

where λ_{CE}, λ_{S} and λ_{T} are the hyper-parameters that control the relative importance of every loss term and $\mathcal{D}$ the set of 1-Lipschitz functions.

Implementation Details

Our model's encoder Φ_{E} is a ResNet 50 [179] pretrained on ImageNet [36] on the task of image labeling. In order to obtain our final model, we divide the training process in two steps. First, our model is trained only using the supervised loss $\mathcal{L}_{\text{CE}}$ to obtain $\Phi^{\star} = \min_{\Phi} \mathcal{L}_{\text{CE}}$ for 6 epochs on YouTube-VOS [69]. As a result, Φ^{*} has an initial understanding of the video object segmentation task and provides a better initialization than ImageNet.

Then, we use Φ^{*} as an initialization to train our spatio-temporal model using the loss defined in Eq. (3.19) in DAVIS17 [68] for 40 epochs. When training either with YouTube-VOS or DAVIS17, we consider every pair of video-object as independent samples.

In our experiments, we observe that adding the critics once the model is initialized closer to the final task helps to stabilize training. With the idea to bring the predicted and the ground truth masks distributions in the discriminators closer at each iteration, we overwrite the ground truth pixel values $\mathbf{Y}_t$ with the values of the predicted masks $\hat{\mathbf{Y}}_t$ that are correctly estimated with an uncertainty lower than 0.25. Also, at each iteration, the ground truth masks are augmented by adding Gaussian noise with mean and variance equal to $\hat{\mathbf{Y}}_t$ statistics.

Our model is trained on images of size 512×512 augmented with horizontal flipping, random scaling with factors $[0.75, 1.25]$ and $[-30, 30]$ degrees rotations. Also, at each training iteration, the reference object frame $\mathbf{M}_n$ of a sequence $\mathbf{x}$ is randomly chosen (instead of always being the first frame in which the object appears). We use Adam [38] with a learning rate 1e-5, $\beta_1 = 0.5$, $\beta_2 = 0.999$, batch size 6 and polynomial decay with power 0.9. During the training of the spatio-temporal model, the learning rate is constant for the first 10 epochs and Φ is optimized once for every 5 optimization steps of the critic networks. The weight coefficients for the loss terms in Eq. (3.19) are set to $\lambda_{\text{CE}} = 100$, $\lambda_{\text{S}} = 1$, $\lambda_{\text{T}} = 1$ and $\lambda_{\text{gp}} = 10$.

In order to better approximate the mask error propagation that occurs at test time during training, we set the temporal window size K to the highest value that fits in our GPU memory, $K = 4$. Note that K is just used during training and only information from the previous frame is used at test time. This parameter is similar to *back propagation through time* introduced in [82] where it was shown that training robustness improves by propagating as many K estimated masks as possible rather than the ground-truth.

We concentrate all computational load to the training stage, which requires 4 NVidia® Titan Xp , 3 of them used for training the regressor and 1 for the critics. Our model takes 2 days to finish pretraining on YouTube-VOS and 3 days for the final training on DAVIS17. During test, we only require one single GPU with at least 600Mb of RAM. When using an NVidia® Titan Xp, we

can process videos up to 32 FPS.

3.4.4 Experimental Evaluation

We thoroughly evaluate our method, FaSTGAN, quantitatively and qualitatively. We compare our approach against current state of the art on semi-supervised video object segmentation: PReMVOS [174], OSVOS^{S} [77], RVOS [96], CRN [93], MoNet [94], RGMP [81], MaskRNN [82], FEELVOS [86], PML [83], OSMN [80], STCNN [180], RVOS [96] and BVS [72].

We evaluate our method on the tasks of single object (DAVIS16 [67]) and multiple object video segmentation (DAVIS17 [68], YouTube-VOS [69]). The segmentation accuracy is reported as region similarity (intersection over union $\mathcal{J}$), contour accuracy ($\mathcal{F}$ measure), and their mean ($\mathcal{J}\&\mathcal{F}$). For the subsets whose annotation is non-public, we compute the results using the submission website provided by the organizers of the challenge.

Ablation Study

In Table 3.3, we perform a comprehensive ablation study to analyze the effect of the different loss components that we use in our method during training. In our baseline, we only use the balanced binary cross entropy loss $\mathcal{L}_{\text{CE}}$ that penalizes wrong predictions at each pixel and frame independently. Therefore, we do not enforce any spatial or temporal consistency.

First, we introduce the spatial discriminator with its associated loss ($\mathcal{L}_{\text{S}}$). This improves substantially the accuracy of the method in the contours of the objects boosting the $\mathcal{F}$ measure by more than 1.5 points. This improvement can also be seen qualitatively in Fig. 3.10 comparing (b) to (c).

After that, we replace the previous discriminator by the temporal one with its associated loss ($\mathcal{L}_{\text{T}}$). As a consequence, performance increases considerably gaining 1.5 points in $\mathcal{J}\&\mathcal{F}$. Now, the model predicts masks with better temporal consistency as can be seen in Fig. 3.10 when comparing the right arm of the man in (Fig. 3.10-c) versus (Fig. 3.10-d). However, the temporal smoothness of the masks enforced by the model is sometimes too severe and the regressor has difficulties recovering from large disoccluded parts (Fig. 3.10-d).

Finally, we combine the spatial and temporal discriminators and their respective losses. As a result, FaSTGAN is capable of incorporating the accuracy improvements in the contours introduced by the spatial discriminator together with the temporal masks propagation smoothness introduced by the temporal discriminator achieving a final score of 81.9 $\mathcal{J}\&\mathcal{F}$ in DAVIS16. As it can be seen in Fig. 3.10-e, the final mask predicted by our spatio-temporal model segments properly the right hand of the man and it can also recover the segmentation of the legs that were occluded in previous frames.

			DAVIS16 Val		
$\mathcal{L}_{CE}$	$\mathcal{L}_S$	$\mathcal{L}_T$	$\mathcal{J}\&\mathcal{F}$	$\mathcal{J}$	$\mathcal{F}$
✓	-	-	80.0	79.8	80.2
✓	✓	-	80.5	79.2	81.8
✓	-	✓	81.5	**80.7**	82.2
✓	✓	✓	**81.9**	80.2	**83.5**

Table 3.3: **Quantitative ablation study**: Comparison between the different loss components.

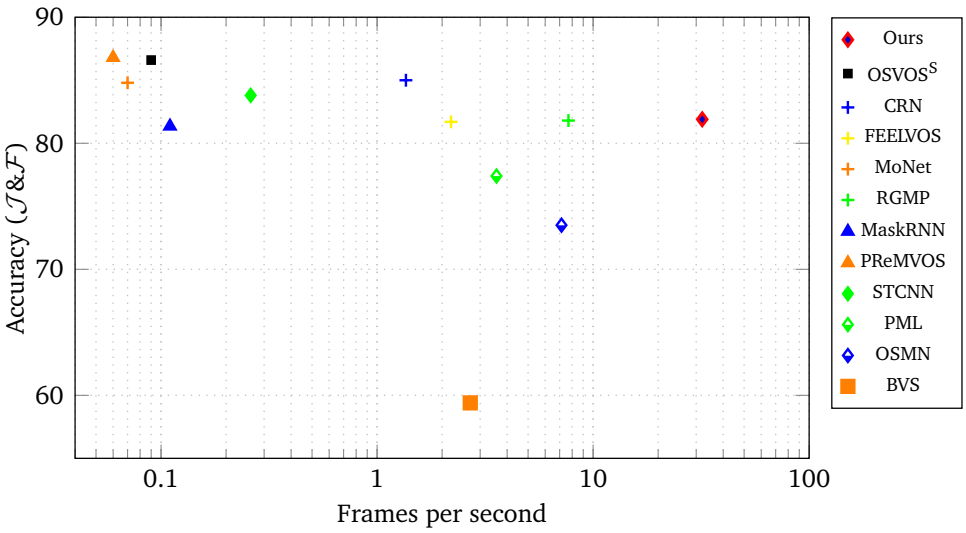


Figure 3.12: **Accuracy versus speed.** $\mathcal{J}\&\mathcal{F}$ in DAVIS16 with respect to FPS.

Measure		Ours	RGMP	OSMN	PML	FEELVOS	CRN	STCNN	MaskRNN	OSVOS^S	MoNet	PReMVOS	RGMP*
$\mathcal{J}\&\mathcal{F}$	Mean $\mathcal{M}\uparrow$	81.9	81.8	73.5	77.4	81.7	85.0	83.8	81.3	86.6	84.8	**86.8**	79.0
FPS $\downarrow$		**32.5**‡/32.2*/30.3†	7.7*	7.1$^{\diamond}$	3.6	2.2	1.4†	0.25	0.11	0.09†	0.07†	0.06	32.2*
$\mathcal{J}$	Mean $\mathcal{M}\uparrow$	80.2	81.5	74.0	75.5	81.1	84.4	83.8	80.4	**85.6**	84.7	84.9	78.4
	Recall $\mathcal{O}\uparrow$	94.6	91.7	87.6	89.6	–	**97.1**	96.1	96.0	96.8	96.8	96.1	92.1
	Decay $\mathcal{D}\downarrow$	9.6	10.9	9.0	8.5	–	5.6	4.9	4.4	5.5	6.4	8.8	**3.6**
$\mathcal{F}$	Mean $\mathcal{M}\uparrow$	83.5	82.0	72.9	79.3	82.2	85.7	83.8	82.3	87.5	84.8	**88.6**	79.7
	Recall $\mathcal{O}\uparrow$	94.3	90.8	84.0	93.4	–	95.2	91.5	93.2	**95.9**	94.7	94.7	90.8
	Decay $\mathcal{D}\downarrow$	9.1	10.1	10.6	7.8	–	5.2	6.4	8.8	8.2	8.6	9.8	**3.6**

Table 3.4: **DAVIS16 benchmark**: FaSTGAN versus the most recent state of the art, more methods can be found in the DAVIS website[3]. RGMP* is pretrained on YouTube-VOS instead of simulated data. Frames per Second (FPS) reported on a Titan X for †, Titan Xp for ‡, Quadro M600 for ⋄ or 1080Ti for ⋆, methods without specifier did not report hardware in their publications.

Evaluation on Single-Object VOS

Fig. 3.12 shows the accuracy using $\mathcal{J}\&\mathcal{F}$ in DAVIS16 versus the frame rate for various state-of-the-art methods. We can clearly see that our method is in a unique position achieving similar accuracy to previous methods that focused on speed like RGMP, OSMN or PML, while improving on their frame rate by at least a factor of 4. When compared to methods that aim to maximize the accuracy (OSVOS^S, MoNet or PReMVOS), we lose around 5 points in $\mathcal{J}\&\mathcal{F}$, but in exchange we increase on their frame rate by a factor of at least 350. As a result, FaSTGAN sets new state of the art in terms of high frame rate while obtaining high accuracy. This brings the field of video object segmentation closer to applications in real time scenarios.

Table 3.4 reports the results presented in Fig. 3.12 quantitatively and in more depth. In general, methods usually focus either on accuracy or speed at the expense of achieving lower performance in the other. Our method clearly focuses on speed while trying to retain as much accuracy as possible. Compared to the best previous method in the fast speed spectrum, RGMP, we achieve a similar accuracy while improving their frame rate by almost a factor of 4 when tested in the same hardware (32.2 vs 7.7 FPS). This improvement is mainly achieved by simplifying their multiscale testing approach by using only a single scale with an image resolution of 512x512. Note that by testing RGMP with our single scale strategy, a similar frame rate is obtained but their accuracy drops by 5 points in $\mathcal{J}\&\mathcal{F}$ (76.78 vs 81.9).

In order to show the effect of YouTube-VOS pretraining in previous methods, we train RGMP by substituting their synthetic data generation pretraining with YouTube-VOS video sequences, denoted as RGMP* in Table 3.4. To do so, we first train their method on YouTube-VOS and then we fine-tune it on DAVIS17, we stop training in both cases when the loss flattens. In order to provide a fair comparison, our single scale test strategy is used, which also improves their FPS. Pretrainig on YouTube-VOS improves their performance by roughly 2 points in $\mathcal{J}\&\mathcal{F}$ (79.0 vs 76.78) which is sill significantly lower than the accuracy of our model.

Evaluation on Multi-Object VOS

For completeness, even though our method is designed for single-object video sequences, we report its performance in multi-object scenarios by considering every pair of video-object as a different video and by sequentially combining the predicted masks.

In Table 3.5, we compare against state-of-the-art methods in DAVIS17 and YouTube-VOS. Our method is again the fastest and it achieves remarkably good accuracy compared to methods trained with multiple objects at the same time. When compared to RGMP pretrained in YouTube-

[3] https://davischallenge.org/davis2016/soa_compare.html
[4] https://davischallenge.org/davis2017/soa_compare.html

	DAVIS17 Val				YouTube-VOS Val			
	$\mathcal{J}\&\mathcal{F}$	FPS	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{J}\&\mathcal{F}$	FPS	$\mathcal{J}$	$\mathcal{F}$
Ours	60.2	**16.5**‡/16.3*/15.4†	57.6	63.4	52.0	**17.2**‡/17.1*/16.1†	50.0	53.9
RVOS	60.6	7.57	57.5	63.6	56.8	7.91	54.6	59.1
OSMN	54.8	3.62$^{\diamond}$	52.5	57.1	51.8	3.79$^{\diamond}$	50.3	52.1
RGMP	66.7	2.97*	64.8	68.6	52.8	2.66*	51.4	54.2
FEELVOS	71.5	1.96	69.1	74.0	–	–	–	–
STCNN	61.7	0.13	58.7	64.6	–	–	–	–
OSVOS^{S}	68.0	0.05†	64.7	71.3	–	–	–	–
PReMVOS	**77.8**	0.03	**73.9**	**81.8**	**72.2**	0.03	**69.3**	**75.2**
RGMP*	56.2	16.3*	52.8	59.6	46.3	17.1*	44.3	48.2

Table 3.5: **DAVIS17 and YouTube-VOS**: FaSTGAN versus the state of the art, more methods can be found in the DAVIS website[4]. FPS is computed assuming linear scaling with the number of objects, thus using FPS from DAVIS16 and multiplying by the mean number of objects in a certain set. Specifier ($*$, †, ‡, $\diamond$, $\star$) definitions are the same than in Table 3.4.

VOS and tested with our single scale strategy, for the same speed, we outperform their model by 4 and 5.7 points in $\mathcal{J}\&\mathcal{F}$ in both DAVIS17 and YouTube-VOS, respectively.

Compared to the original RGMP model, our method has a similar accuracy in YouTube-VOS while running almost 6.5 faster. In DAVIS17, there are several small objects and their multi-scale testing strategy helps to obtain better performance in such scenarios. As a result, FaSTGAN with a single-scale strategy achieves a slightly worse result but it runs more than 5 times faster than RGMP.

Overall, our method is the only one running at real-time ($>$ 12 FPS), enabling video object segmentation applications on the fly.

Fairness in method comparison

We would like to briefly discuss the difficulties in providing a fair comparison with other video object segmentation models. First of all, methods in Table 3.4 and Table 3.5 have been pre-trained in a wide variety of different datasets. For instance, OSVOS^{S}, OSMN, PML and PReMVOS use COCO [181]; MoNet, PReMVOS, PML, RGMP and CRN use PASCAL VOC [182, 183]; and RGMP uses as well ECSSD [184] and MSRA 10K [185]. Also, MoNet, PReMVOS, and CRN use optical flow obtained from Flownet2.0 [186] which is trained using [187, 188]. Before the release of YouTube-VOS, static image datasets were used to train most methods due to the lack of a large scale video object segmentation dataset. We expect future methods to gradually converge to YouTube-VOS pretraining which would enable an easier comparison among methods.

Moreover, previous methods report timings in a wide variety of GPU types. In order to provide a fair comparison, we list the GPU type used in each publication in Table 3.4 and

Figure 3.13: **Qualitative results**: Sample sequences from YouTube-VOS (top 2), DAVIS17 (middle 2) and DAVIS16 (bottom 2). In each row, the leftmost image is the initial reference frame, the rest of the images are the predictions for the following frames.

Table 3.5 when available and we test our method in the three different GPU types that we have at our disposal, 1080Ti, Titan Xp and, Titan X.

Qualitative Results

Fig. 3.13 shows examples of the predicted masks using our approach, FaSTGAN. The first column displays the reference mask $\mathbf{M}_n$ and the rest of the columns display the segmented mask by our method in the following frames.

Note that even when the input frame is corrupted by occlusions, changes of appearance and dynamic background, our method remains robust.

3.5 Non-Rigid Surface Reconstruction

Once we can segment objects in a scene, we are particularly interested in reconstructing them. Motivated by the current success of Deep Learning methods for estimating depth maps from single images of a scene [189–191], in this section we tackle the related problem of estimating the underlying parametric model defining the shape of a non-rigid surface from a single image. This problem has been traditionally addressed in the context of the Shape-from-Template (SfT) [105], requiring a reference template image of the surface for which the 3D geometry is known, and a set of 3D-to-2D point correspondences or a mapping between this template and the input image. This approach, however, may be difficult to hold in practice, specially when considering low-textured surfaces.

We relax previous assumptions and present a learning-based approach that allows for globally non-rigid surface reconstruction from a single image without relying on point correspondences, and which in particular, shows robustness to situations rarely addressed previously: lack of surface texture and large occlusions. Our model is based on a fully differentiable Deep Neural Network that estimates a 3D shape from a single image in an end-to-end manner, and builds upon three branches that enforce geometry consistency of the solution.

More exactly, as illustrated in Fig. 3.14, a first branch of the proposed architecture (the *2D detection branch*) is responsible for localizing the mesh onto the image, and for fitting a 2D grid to it. The 2D vertices of this grid are then lifted to 3D by a *depth branch*, a regressor that combines the 2D detector confidence maps and the input image features. Finally, a *shape branch* is responsible for recovering the full shape while ensuring that the estimated 3D coordinates correctly re-project onto the image. During training, this branch also incorporates a novel fully-differentiable layer that performs a Procrustes transformation and aligns the estimated 3D mesh with the ground truth one. This branch is important as it was proven important to perform Procrustes alignment in previous approaches for adapting to datasets with different reference frames and metrics. It also favors convergence of the learning process.

Since there is no dataset large enough to train data-hungry Deep Learning algorithms such as ours, we have created our own using a rendering tool. We have synthesized 128,000 photo-realistic pairs input 2D-image/3D-shape accounting for different levels of deformations, amount and type of texture, material properties, viewpoints, lighting conditions and occlusion. Fig. 3.16-top shows some examples. Evaluation on a test split of this dataset demonstrates remarkable improvement of our network compared to state-of-the-art SfT techniques, which typically rely on known 3D-to-2D correspondences, especially under strong occlusions and poorly-textured surfaces. Furthermore, our model learned with synthetic data can be easily fine-tuned to real sequences, using just a few additional real training samples. Results on the CVLab sequences [2]

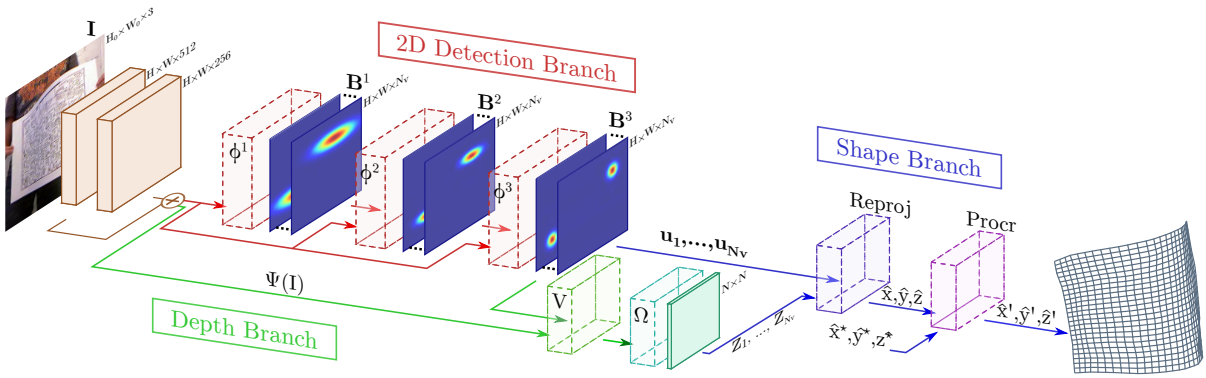


Figure 3.14: **DeformNet overview.** The proposed architecture consists of three main branches. The 2D *detection branch* is responsible for the 2D location of the mesh and the associated belief maps. The *depth branch* lifts the 2D detected mesh by leveraging on image cues and the detection uncertainties. Finally, the *shape branch* fuses the 2D detections and their estimated depths to obtain 3D shape in such a way that perspective projection is enforced. An additional *procrustes layer* is used during training to align the estimated mesh with the ground truth one.

with a bending paper and a deforming t-shirt clearly show that our method outperforms existing approaches.

In summary, this section main contributions are: 1) the first—to the best of our knowledge—fully-differentiable model for non-rigid surface reconstruction from a single image that does not require initialization, accurate knowledge of the template, 3D-to-2D correspondences, nor hand-crafted constraints; 2) a geometry-aware architecture that embeds a pinhole camera model and encodes rigid alignment during training; and 3) a large photo-realistic publicly available dataset of images of non-rigid surfaces annotated with the corresponding 3D shapes, which we hope it will inspire future research in the field.

3.5.1 DeformNet Dataset

It is well known that deep networks require large amounts of training data. However, the only existing dataset we are aware of that contains non-rigid surfaces annotated with ground-truth 3D shape is [2], which includes 505 images of a bending paper and a deforming t-shirt. This is far below what is needed, specially if we expect our network to generalize to non-observed textures. For this purpose, we have created a large synthetic dataset with 128,000 samples rendered with Autodesk™- Maya. Each sample consists of a 224×224 image and a 3D shape represented by a 9×9 triangular mesh. A few examples of the dataset are shown in Fig. 3.16-top.

We generated our dataset by varying textures, deformations and lighting conditions. Con-

cretely, we have chosen 200 different textures from [192] which is formed by repetitive patterns, rich, poor and plain textures. The deformations were generated for 40 different meshes (same topology but varying aspect ratios and sizes). The mesh dynamics were rendered by simulating a hanging piece of material held with up to 4 pins and moving with the wind. Four different materials, defined with four different stiffness matrices, were considered. The scene was lit by one point light source of high intensity with a random position, plus a component of ambient illumination. In all cases, we assumed a Lambertian reflectance.

The rendered dataset was augmented with all three possible flips of each image. Additionally, for each image, three new ones were generated by applying a random rigid transformation on the corresponding deformable surface. At training time, the dataset was further augmented with random color changes at pixel level (hue, saturation, contrast and brightness). The dataset is publicly available at https://www.albertpumarola.com/research/DeformNet.

3.5.2 Problem Formulation

We aim at designing a Deep Learning framework that directly estimates a non-rigid 3D shape from an input RGB image $\mathbf{I} \in \mathbb{R}^{H_o \times W_o \times 3}$. The shape is represented as a triangulated 3D mesh with N_v vertices $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_{N_v})$, where $\mathbf{x}_i = (x_i, y_i, z_i)$ are the coordinates of the i-th vertex, expressed in the camera coordinate system. In the following, we assume the structure of the mesh to be known, being a $N \times N$ rectangular grid, *i.e.*, $N_v = N^2$.

We also assume the calibration parameters of the camera to be known, namely the focal lengths, f_u and f_v, and the principal point (u_c, v_c).

3.5.3 Geometry-aware Network

Our framework for estimating a non-rigid shape from a single image is shown in Fig. 3.14. We have devised an architecture with three branches, each responsible of reasoning about a different geometric aspect of the problem. The first two branches are arranged in parallel and perform probabilistic 2D detection of the mesh in the image plane and depth estimation (red and green regions in the figure, respectively). These two branches are then merged (blue region in the figure) in order to lift the 2D detections to 3D space, such that the estimated surface correctly re-projects onto the input image and it is properly aligned with the ground truth shape. In the results subsection we will show that reasoning in such a structured way provides much better results than trying to directly regress the shape from the input image, despite using considerably deeper networks.

2D Detection Branch

Given an input image $\mathbf{I}$, the first step consists in extracting image features from a pre-trained network, in our case we concatenate two Resnet V2 blocks [193]. For each block, the stride of the last unit is set to one, in order to keep the same spatial resolution for the two units. Let us denote these features as $\Psi(\mathbf{I}) \in \mathbb{R}^{H \times W \times C}$ being H, W and C the height, width and depth features dimensions.

The image features are then fed into the 2D detection network, which is responsible for estimating the 2D locations of the mesh vertices $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_{N_v}) \in \mathcal{U}$, where $\mathbf{u}_i = (u_i, v_i)$ and $\mathcal{U}$ is the set of all (u, v) pixel locations in the input image $\mathbf{I}$. Drawing inspiration on the convolutional pose machines [194] for human pose estimation, the 2D location of each vertex $\mathbf{u}_i$ is represented as a probability density map $\mathbf{B}_i \in \mathbb{R}^{H \times W}$ computed over the entire image domain as:

$$\mathbf{B}_i[u, v] = P(\mathbf{u}_i = (u, v)) \ \ , \forall \ (u, v) \in \mathcal{U}. \tag{3.20}$$

As in [194] these belief maps are estimated in an iterative manner. In particular, let $\mathbf{B}^t = (\mathbf{B}_1^t, \ldots, \mathbf{B}_{N_v}^t) \in \mathbb{R}^{H \times W \times N_v}$ be the concatenation of all belief maps at iteration t. This tensor is estimated by a regressor function Φ^t, which takes as input the image features and the concatenated belief maps at the previous stage $t-1$:

$$\Phi^t(\Psi(\mathbf{I}), \mathbf{B}^{t-1}) \rightarrow \mathbf{B}^t \ . \tag{3.21}$$

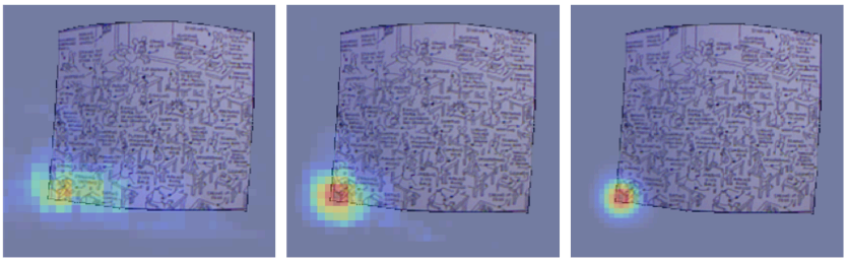
In the first step, the regressor is only fed with the image features, that is $\Phi^1 \equiv \Phi^1(\Psi(\mathbf{I}))$. We denote by T_{max} the maximum number of iterations. As it is shown in Fig. 3.15, after each iteration, the location of the vertices is progressively refined.

In order to implement the regressor $\Phi^t(\cdot)$ we use again ResNet V2 blocks followed by two convolutional layers. The output of each Φ^t is normalized with respect to H and W to guarantee that $\sum_{u=1}^{H} \sum_{v=1}^{W} \mathbf{B}_i^t[u, v] = 1$, $\forall i \in \{1, \ldots, N_v\}$, and $\forall t \in \{1, \ldots, T_{\text{max}}\}$.

Finally, it is worth noting that the 2D detection branch we have just described is fully differentiable. The output $\mathbf{u}_i = (u_i, v_i)$ for the $i-$th vertex can be estimated as the following weighted sum over the last belief map $\mathbf{B}^{T_{\text{max}}}$:

$$u_i = \frac{\sum\limits_{(u,v) \in \mathcal{U}} u \cdot \mathbf{B}_i^{T_{\text{max}}}[u, v]}{\sum \mathbf{B}_i^{T_{\text{max}}}}, \qquad v_i = \frac{\sum\limits_{(u,v) \in \mathcal{U}} v \cdot \mathbf{B}_i^{T_{\text{max}}}[u, v]}{\sum \mathbf{B}_i^{T_{\text{max}}}}$$

where $\sum \mathbf{B}_i^{T_{\text{max}}}$ sums over all elements of $\mathbf{B}_i^{T_{\text{max}}}$. These 2D estimates will be forwarded to the shape branch' previously described, while the belief maps in $\mathbf{B}^{T_{\text{max}}}$ will be used to infer the depth



$t = 1$ $t = 2$ $t = 3$

Figure 3.15: **Refinement of the 2D vertices position.** Output (for one specific vertex) of the regressor Φ^t for three consecutive time steps. Note how the uncertainly in the vertex location is progressively reduced.

value for each of the vertices in the depth branch.

Depth Branch

The belief maps $\mathbf{B}_i^{T_{\max}}$ of the 2D vertex locations are forwarded to the depth branch, to estimate the depth coordinate z_i for every vertex. Note that previous works in related problems like 3D human pose estimation [195, 196] have not taken advantage of the uncertainty typically associated to the feature detectors.

To do so, the proposed layer produces new feature maps $\mathbf{V}(\mathbf{B}^{T_{\max}}, \Psi(\mathbf{I})) \in \mathbb{R}^{N \times N \times C}$, that condition the input feature maps $\Psi(\mathbf{I}) \in \mathbb{R}^{H \times W \times C}$ with the probability maps $\mathbf{B}^{T_{\max}} \in \mathbb{R}^{H \times W \times N_v}$, that is:

$$\mathbf{V}[j(i), k(i), c] = \sum_{(u,v) \in \mathcal{U}} \mathbf{B}_i^{T_{\max}}[u, v] \cdot \Psi(\mathbf{I})[u, v, c] \tag{3.22}$$

$\forall i \in \{1, \ldots, N_v\}, c \in \{1, \ldots, C\}$, where $(j(i), k(i))$ converts the i-th input of an N_v-dimensional vector into a two dimensional input of an $N \times N$ matrix (recall that $N_v = N^2$).

These image features conditioned on the vertices 2D locations are then used as input of a regressor $\Omega(\cdot)$ to estimate the vertices' depth:

$$\Omega(\mathbf{V}(\mathbf{B}^{T_{\max}}, \Psi(\mathbf{I}))) \rightarrow (z_1, \ldots, z_{N_v}). \tag{3.23}$$

Again, the regressor $\Omega(\cdot)$ consists in two ResNet V2 blocks followed by two convolutional layers and the full branch (conditioned features + regressor) is fully differentiable.

Shape Branch

The 2D locations and depth estimates are merged in order to estimate the shape while enforcing the projection constraints and rigid alignment consistency.

Given the estimates (u_i, v_i, z_i) in Eqs. (3.22) and (3.23) of the two first branches, the 3D position $\mathbf{x}_i = (x_i, y_i, z_i)$ of each vertex is recovered with a differentiable layer that models the pinhole reprojection model:

$$x_i = z_i \cdot \frac{u_i - u_c}{f_u}, \qquad y_i = z_i \cdot \frac{v_i - v_c}{f_v}, \qquad z_i = z_i \,. \tag{3.24}$$

This gives us an estimate of the deformable shape $\mathbf{X}$, and we could train the network by considering the L2 loss $||\mathbf{X} - \mathbf{X}^*||_2^2$ where $\mathbf{X}^*$ is the ground truth 3D shape. However, we propose introducing an additional layer, which computes the Procrustes alignment error between $\mathbf{X}$ and $\mathbf{X}^*$ in a fully differentiable manner, and build our loss function based on this error. Although this layer is removed at test time, we observed that it favors the convergence during training, helps adapting to different datasets, and most importantly, it improves the capacity of the rest of the network to capture the non-rigid component of the shape.

The Procrustes layer ('Procr' box in Fig. 3.14) is implemented by first normalizing $\mathbf{X}$ and $\mathbf{X}^*$ with respect to translation and scale. Let us denote by $\hat{\mathbf{X}} = (\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_{N_v})$ and $\hat{\mathbf{X}}^* = (\hat{\mathbf{x}}_1^*, \ldots, \hat{\mathbf{x}}_{N_v}^*)$ these normalized versions.

Following [197], we can then compute the alignment error between $\hat{\mathbf{X}}$ and $\hat{\mathbf{X}}^*$, without having to explicitly estimate their relative rotation and translation as follows:

$$\mathcal{L}_a(\hat{\mathbf{X}}, \hat{\mathbf{X}}^*) = \sqrt{\frac{\sum_{i=1}^{N_v} |\hat{\mathbf{x}}_i|^2 + |\hat{\mathbf{x}}_i^*|^2 - 2\lambda_{\text{max}}}{N_v}} \tag{3.25}$$

where λ_{max} is the maximum eigenvalue of a 4×4 matrix built in terms of the elements of $\hat{\mathbf{X}}$ and $\hat{\mathbf{X}}^*$. Since there exist differentiable approximations of the eigendecomposition (for example, the function `tf.self_adjoint_eigvals` in Tensorflow), the full shape branch is again differentiable.

3.5.4 Learning the Model

The cost function that we aim to minimize is a combination of the 3D alignment error in Eq. (3.25) and the 2D detection error produced at the output of each regressor Φ^t, for $t = \{1, \ldots, T_{\text{max}}\}$:

$$\mathcal{L} = \mathcal{L}_a(\hat{\mathbf{X}}, \hat{\mathbf{X}}^*) + \gamma \sum_{t=1}^{T_{\text{max}}} \|\mathbf{B}^t - \mathbf{B}^*\|_2^2 \,, \tag{3.26}$$

where $\mathbf{B}^*$ is a heat-map generated by placing Gaussian peaks at the ground truth 2D locations (u_i^*, v_i^*) of the mesh vertices. γ denotes a weight used to give similar orders of magnitude to each of the terms of the loss function.

Method	Known Text	New Text	No-Text	Time (ms)
Ba15Iso	8.54 / -	8.72 / -	- / -	495
Ba15Iso-It	5.65 / -	6.78 / -	- / -	15,507
Ba15Conf	30.50 / -	31.91 / -	- / -	11,232
Ch14IsoLsq	6.74 / -	6.95 / -	- / -	2618
Ch14IsoLsq-It	4.85 / -	5.3 / -	- / -	14,813
Resnet-50 V2	**0.92** / **3.83**	11.23 / 18.50	8.39 / 9.43	**152**
DeformNet	2.64 / 4.57	**3.28** / **4.09**	**2.86** / **4.62**	219

Table 3.6: **Evaluation on synthetic data.** Euclidean average distance between 3D ground-truth and estimated 3D reconstruction. Each pair 'err1 / err2' indicates the error without and with occlusions, respectively. Execution time in the last column is computed as the average time (in ms) to reconstruct a sample. Symbol '-' indicates that the method was not evaluated on this scenario, as they correspond to situations (no texture or large occlusions) that can not be addressed by template-based analytical solutions.

Implementation Details

The model is trained with the synthetically generated dataset described in the next section, made of $H_o \times W_o = 224 \times 224$ images. The image features $\Psi(\mathbf{I})$ are obtained from a Resnet V2 network pre-trained on ImageNet, resulting in feature maps of size $H \times W \times C = 56 \times 56 \times 768$. In all our experiments we consider meshes of spatial resolution $N \times N = 9 \times 9$, thus, $N_v = 81$. The resulting belief maps B^t will be therefore of size $56 \times 56 \times 81$. In the 2D detection branch, we fixed the maximum number of iterations to $T_{\text{max}} = 3$, as further stages did barely change the resulting belief maps distributions.

The training procedures is split in two stages: initially, only the regressors Φ^t are trained. Then, regressors Φ^t and Ω are jointly trained. In both cases, the parameters of the feature extractor $\Psi(\mathbf{I})$ are kept fixed. In Eq. (3.26) we set $\gamma = 5 \cdot 10^{-3}$. We use Adam solver [38] with a batch size of 3 images and weight decay of $4 \cdot 10^{-5}$. Every 2 epochs we exponentially decay the learning rate, which is initially set to $2 \cdot 10^{-4}$.

3.5.5 Experimental Evaluation

We now present results on synthetic and real data. We compare our approach, which we dub *DeformNet*, with the following state-of-the-art template-based solutions: Ba15Iso, the isometry-based solution proposed in [105]; Ba15Conf, a conformal-based approach, also from [105]; Ch14IsoLsq, the least-squares isometric reconstruction of [106]. We denote by Ba15so-It and Ch14IsoLsq-It the same previous methods after executing 25 iterations of the non-linear refinement proposed in [198]. This refinement step could not be applied to Ba15Conf due to computational time constraints. [106] showed that Ch14IsoLsq-It systematically outperformed

the same baselines we consider here and also the methods introduced in [110, 111, 198]. We therefore consider Ch14IsoLsq-It to be the best current analytic approach to assess the potential of our solution. Additionally, we also compare against a deep network baseline, consisting of a ResNet-50 V2 architecture [193] directly inferring 3D mesh coordinates.

In the following, we will report the reconstruction error, computed as the L2 distance between the estimated and the ground truth shapes (dimensionless for the synthetic results and in mm for the real ones). As common practice, the estimated meshes are aligned to the ground truth before evaluation using a Procrustes transformation. Additionally, in order to make a fair comparison, all methods requiring the pixels coordinates of the mesh, are fed with the estimates $\mathbf{U} = (\mathbf{u}_1, \ldots, \mathbf{u}_{N_v})$ obtained with our network, augmented to a few hundreds of template-to-image correspondences by interpolation. We would like to point that our network produces an error of approximately 2 pixels in these 2D detections, and computing them using feature descriptors such as SIFT [199], generally led to worse results as these type of descriptors are prone to fail for non-textured surfaces with repetitive pattens and self-occlusions.

Evaluation on Synthetic Data

We evaluated all methods on a test set of our dataset consisting of 1208 independent samples generated with random values of shape and camera pose. These test samples are split into three subsets: 553 unknown shapes with a texture seen at training time ('Known Texture'), 553 unknown shapes with a texture not seen at training time ('New Texture'), and 102 unknown shapes without texture or very poorly textured ('Non-Textured'). Additionally we have simulated occlusions by covering the input images with a number of gray rectangular patches randomly distributed. Examples of the type of input images for each test case are shown in Fig. 3.16-top.

Template-based analytical methods (Ba15Iso, Ba15Conf, Ch14IsoLsq and their iterative versions) were only evaluated on the textured and non-occluded cases, as they are methods that by construction can not realistically address the lack of texture or strong occlusions. Alternatively, to make the learning approaches (Resnet-50 V2 and DeformNet) robust to occlusions, the two networks were retrained with the gray-patched images. No retraining was done to handle the lack of texture.

Table 3.6 summarizes the results of the synthetic evaluation. When dealing with textured and non-occluded images, Ch14IsoLsq-It is, as expected, the most accurate solution among the analytical methods. Regarding the learning approaches, Resnet-50 V2 turns to work very well under known textures. However, its performance suffers a big drop when dealing with textures not seen during training and with poorly textured surfaces. DeformNet performs consistently well in all situations, outperforming in all cases the analytical solutions. Particularly interesting is the case when dealing with new textures that are occluded, in which we obtain an accuracy

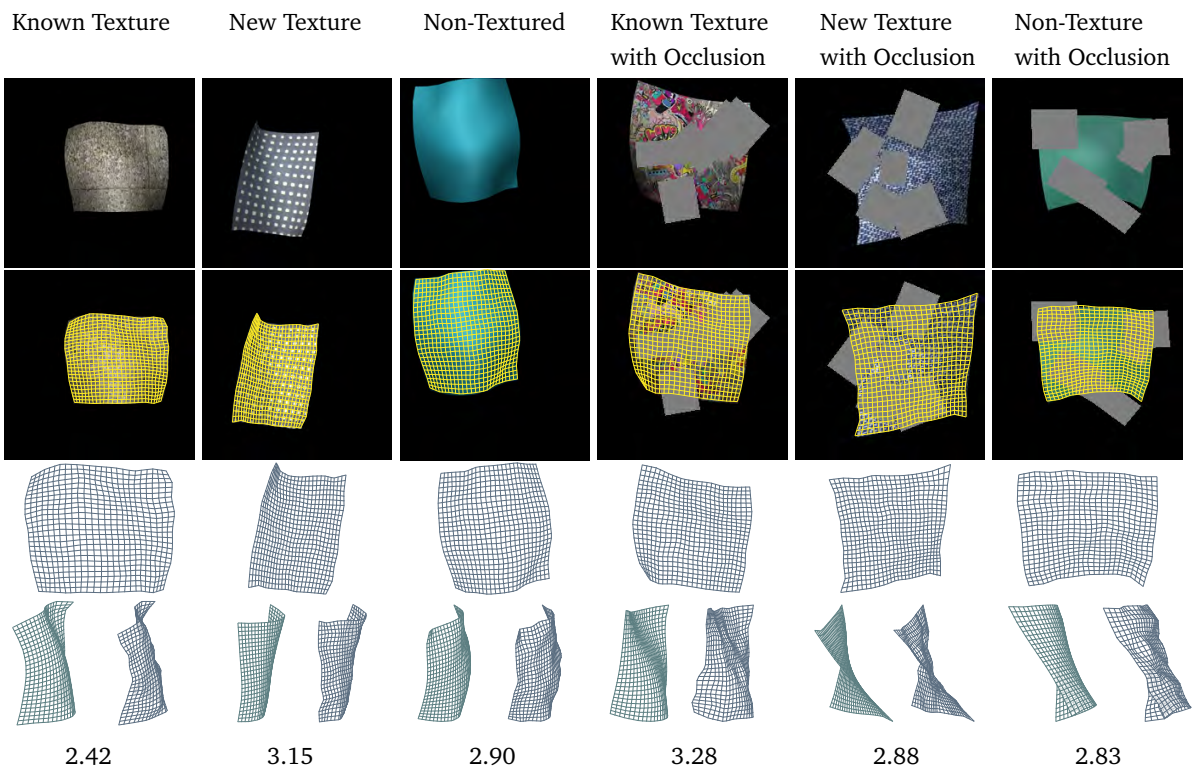
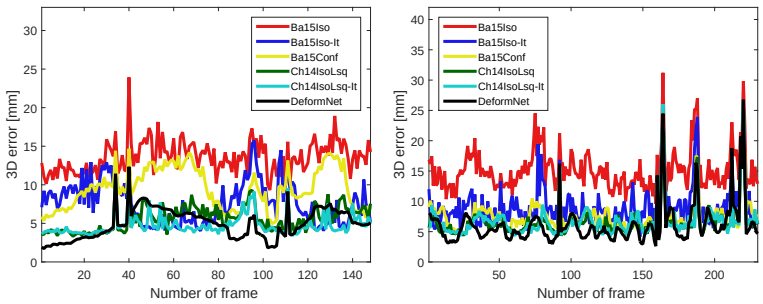


Figure 3.16: **Results on synthetic data.** Reconstructions samples in each of the six cases we consider (surfaces with known, new or no-texture, and with and without occlusions). **First Row:** Input image. **Second Row:** 3D estimated mesh projected onto the input image. **Third Row:** 3D estimated mesh seen from the camera view. **Last Row:** Side view of the ground truth mesh and our estimation (green and blue meshes, respectively). The reconstruction error is indicated at the bottom, to give significance to the errors in Table 3.6.

very similar to the best analytical methods (we obtain 3.62mm versus 3.57mm for competing methods) when dense non-occluded correspondences are provided.

Fig. 3.16 shows examples of the reconstructed meshes obtained by our approach. Note that when there are no occlusions, the recovered shape highly resembles the ground truth, even for non-textured surfaces and not previously seen textures. When the input image is corrupted by occlusions, our solutions turn to be noisier, but even in this case, they are very close to the ground truth.

Computation Times. Another advantage of learning based approaches is that once they are learned, they are much faster than the analytical solutions. The last column of Table 3.6 shows that computing the shape can be done in a fraction of a second for either Resnet-50 V2 and our approach, between one and two orders of magnitude faster than analytical methods.



Method	Paper	T-Shirt
Ba15Iso	13.45	15.2
Ba15Iso-It	7.76	9.01
Ba15Conf	9.26	6.67
Ch14IsoLsq-It	5.66	6.66
Ch14IsoLsq-It	4.81	6.71
Resnet-50 V2	16.38	39.04
DeformNet	**4.79**	**6.11**

Figure 3.17: **Evaluation on the CVLab sequences [2].** The two graphs plot the 3D reconstruction error per frame (in mm) for all methods in the two real sequences (Left: Paper bending sequence, Right: T-shirt sequence). The results of Resnet-50 V2 are not plotted as it was not able to generalize to these sequences. Right: Mean reconstruction errors of all methods.

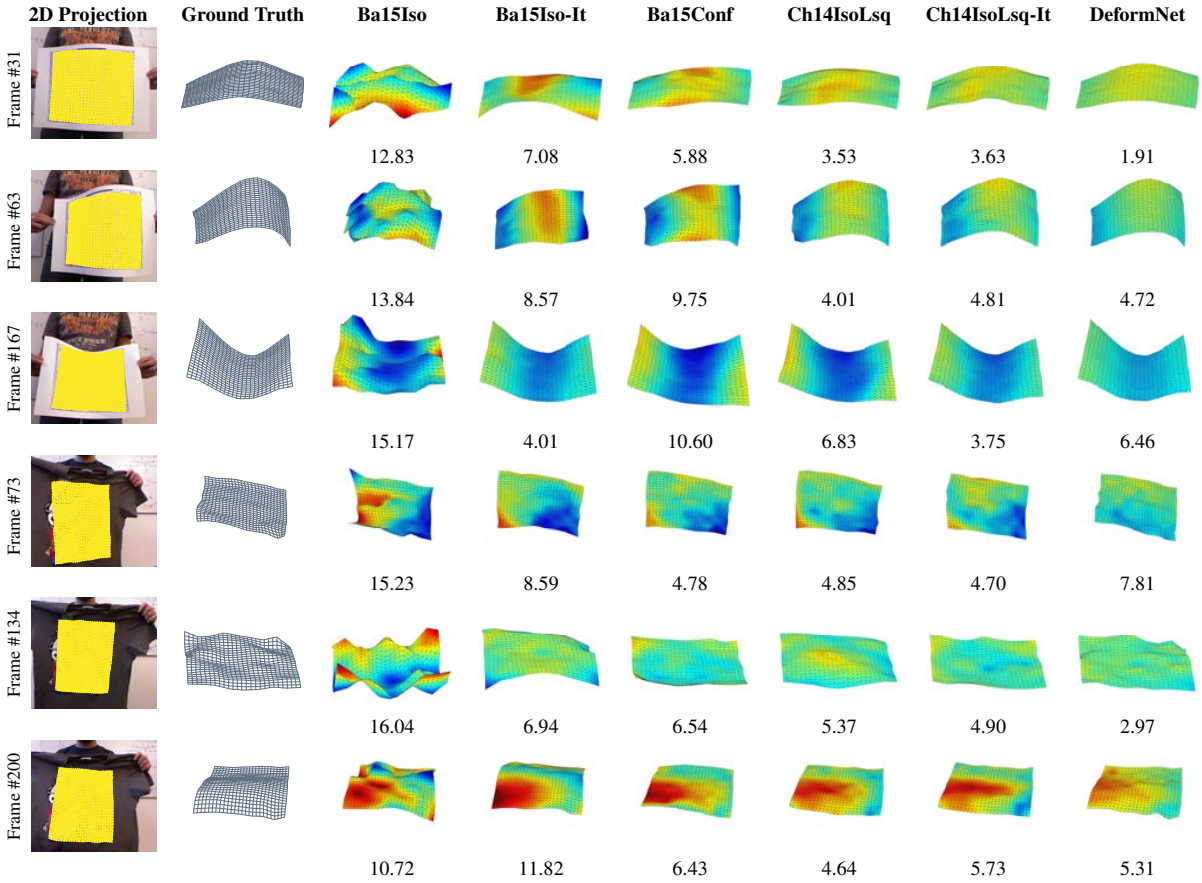


Figure 3.18: **Reconstructed meshes on the 'paper bending' and 't-shirt' CVLab sequences.** Results on Resnet-50 are not included as it did not generalize to real sequences. Each shape is color coded according to its reconstruction error. Larger errors appear in red, and small errors in dark blue. Below each reconstructed shape we indicate the mean reconstruction error (in mm).

Evaluation on Real Data

We also evaluate all methods on two real datasets provided by CVLab [2], which consist in video sequences of a bending paper and a deforming t-shirt, with 193 and 312 frames, respectively. As common practice, the background of the sequences was subtracted. Additionally, both for Resnet-50 V2 and DeformNet, we performed a finetuning of the networks with a very small portion of the dataset (15% first frames). This finetuning was necessary to capture the bounds of the real deformations and adapt to the true illumination conditions that were not rendered by the synthetic dataset. In all methods we evaluated with the rest of the 85% of the frames. Again, for the fairness of comparison, the analytical solutions were fed by the 2D inputs of the mesh obtained by DeformNet, augmented to 500 correspondences using interpolation. The mean 2D location error (in pixels) obtained using DeformNet was 1.24 (paper bending sequence) and 2.28 (t-shirt sequence).

In Fig. 3.17 we plot the 3D reconstruction error per frame for all methods. The table on the right of the figure summarizes the results. Again, our DeformNet is the most accurate approach. In the bending paper sequence the analytic solution of Ch14IsoLsq-It is very close to ours, although DeformNet improves this method by a larger margin in the t-shirt sequence. In any event, recall that DeformNet performs inference per image in a fraction of a second while Ch14IsoLsq-It requires about 15 seconds. For these sequences, Resnet-50 V2, the other Deep Learning baseline we considered, performs very poorly demonstrating that the specific architecture we use in DeformNet allows for a much better generalization.

Finally, Fig. 3.18 shows a few reconstructed shapes obtained for each of the methods. Below each sample, we indicate the reconstruction errors. Note that samples with errors of about 4mm (in the paper bending sequence) or 6mm (in the t-shirt sequence) are already very good solutions. This is the magnitude of the error obtained by DeformNet.

3.5.6 Discussion

One of the most significant aspects of our network is its ability to generalize to unknown textures (see results in Table 3.6). We conjecture that this is the result of two factors: (i) training with a large variety of textures, and (ii) separating the network into two input branches, one for performing 2D detection and the other to modulate input image features using the belief maps of the 2D detections. That is, our two branches allow us to correctly combine appearance and geometry. Note that the Resnet-50 V2 baseline we evaluated was also trained with a variety of textures, but it was not capable to generalize to new textures.

It is well known that on developable surfaces one may reconstruct shape from only the image boundaries [200]. One might therefore think that the robustness of DeformNet to new textures

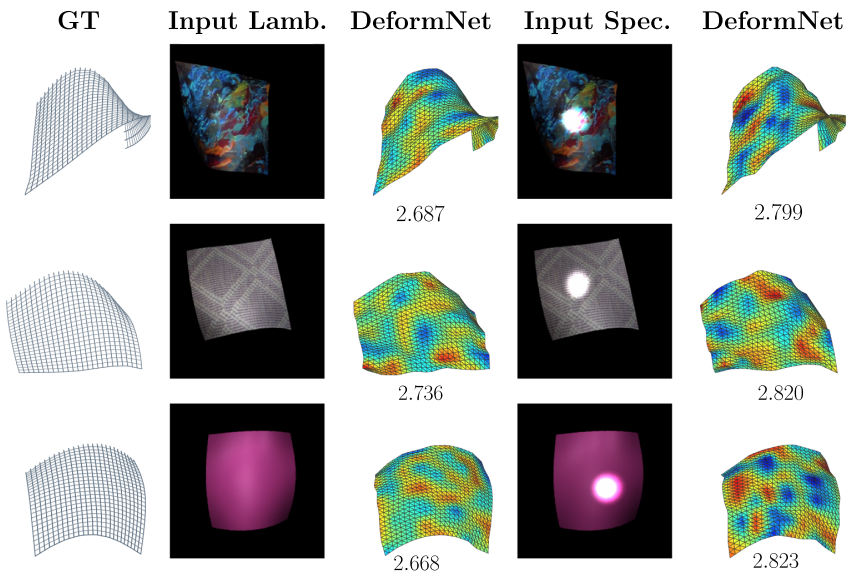


Figure 3.19: **Reconstruction under artificial specularities.** As in Fig. 3.18, each shape is color coded according to its reconstruction error.

might be because our architecture learns to infer shape from the boundaries. In order to evaluate this, we performed the following experiment.

Blurred contours. To reduce the dependency of DeformNet on the contours, we retrained it on a training set in which the surface boundaries of the input images were artificially corrupted by both adding random noise to the 2D coordinates of the boundary vertices and then blurring the contours. This strategy was also used in [201] to evaluate planar homographies. We then tested our architecture on the full dataset and obtained an error of 3.77mm, which is just slightly above the results reported in Table 3.6. Therefore, we can conclude that our network does not highly depend on the boundaries and exploits the whole image.

Relaxing Lambertian reflectance assumptions. To further test our model limits Fig. 3.19 presents an evaluation of the model under synthetic specularities. The network also shows robustness to this scenario, and the overall reconstruction error (2.82) remains very similar to the case with Lambertian assumptions.

3.6 Modeling the Geometry of Dressed Humans

Once we are capable of detecting objects in a scene and performing monocular reconstruction of simple surfaces, we next tackle the more complex task of monocular reconstruction of dressed humans. With the advent of Deep Learning, the problem of predicting the geometry of the human body from single images has experienced a tremendous boost. The combination of CNNs

with large Motion Capture Data (MoCap) [202, 203], resulted in a substantial number of works that robustly estimate the 3D position of the body joints [3, 195, 196, 204–209] or predict their temporal evolution [210–212].

In order to estimate the full body shape, a standard practice adopted in [6, 140, 142, 144, 213, 214] is to regress the parameters of low rank parametric models [138, 215]. Nevertheless, while these parametric models describe very accurately the geometry of the naked body, they are not appropriate to capture the shape of clothed humans.

Current trends focus on proposing alternative representations to the low rank models. Varol *et al.* [158] advocate for a direct inference of volumetric body shape, although still without accounting for the clothing geometry. Very recently, [159] uses 2D silhouettes and the visual hull algorithm to recover shape and texture of clothed human bodies. Despite very promising results, this approach still requires frontal-view input images of the person with no background, and under relatively simple body poses.

In this section, we introduce a general pipeline to estimate the geometry of dressed humans which is able cope with a wide spectrum of clothing outfits and textures, complex body poses and shapes, and changing backgrounds and camera viewpoints. For this purpose, we contribute in three key areas of the problem, namely, the data collection, the shape representation and the image-to-shape inference.

Concretely, we first present 3DPeople (Fig. 3.20) a new large-scale dataset with 2 Million photorealistic synthetic images of people under varying clothes and apparel. We split the dataset 40 male/40 female with different body shapes and skin tones, performing 70 distinct action. The dataset contains 3D geometry of both the naked and dressed body, and additional annotations including skeletons, depth and normal maps, optical flow and semantic segmentation masks. This additional data is indeed very similar to SURREAL [3] which was built for similar purposes. The key difference between SURREAL and 3DPeople, is that in SURREAL the clothing is directly mapped as a texture on top of the naked body, while in 3DPeople the clothing does have its own geometry.

As essential as gathering a rich dataset, is the question of what is the most appropriate geometry representation for a deep network. We consider the Geometry Image proposed originally in [157] and recently used to encode rigid objects in [5, 156]. The construction of the Geometry Image involves two steps, first a mapping of a genus-0 surface onto a spherical domain, and then to a 2D grid resembling an image. Our contribution here is on the spherical mapping. We found that existing algorithms [4, 5] were not accurate, especially for the elongated parts of the body. To address this issue we devise a novel spherical area-preserving parameterization algorithm that combines and extends the FLASH [4] and the optimal mass transportation methods [216].

Our final contribution consists of designing a generative network to map input RGB images

Figure 3.20: **3DPeople dataset.** We present a synthetic dataset with 2 Million frames of 80 subjects (40 female/40 male) performing 70 different actions. The dataset contains a large range of distinct body shapes, skin tones and clothing outfits, and provides 640×480 RGB images under different viewpoints, 3D geometry of the body and clothing, 3D skeletons, depth maps, optical flow and semantic information (body parts and cloth labels). We use the 3DPeople dataset to model the geometry of dressed humans. The dataset can be explored and downloaded at https://cv.iri.upc-csic.es/

of a dressed human into his/her corresponding Geometry Image. Since we consider $128 \times 128 \times 3$ Geometry Images, learning such a mapping is highly complex. We alleviate the learning process through a coarse-to-fine strategy, combined with a series of geometry-aware losses. The full network is trained in an end-to-end manner, and the results are very promising in variety of input data, including both synthetic and real images.

In summary, this section main contributions are: 1) build the first large-scale synthetic dataset and benchmark with 2.5 Million photo-realistic images of 80 subjects performing 70 activities and wearing diverse 3D outfits; 2) a novel representation for 3D meshes that maps 3D data into a 2D space emulating an image, therefore being a 3D data representation compatible with all existing literature on image-based CNN architectures; and 3) a new model that, given an input image of a dressed human in the wild, reconstructs the 3D mesh of the body pose and clothing shape.

3.6.1 3DPeople Dataset

Datasets are fundamental in the deep-learning era. While obtaining annotations is quite straightforward for 2D poses [217–219], it requires using sophisticated MoCap systems for the 3D case.

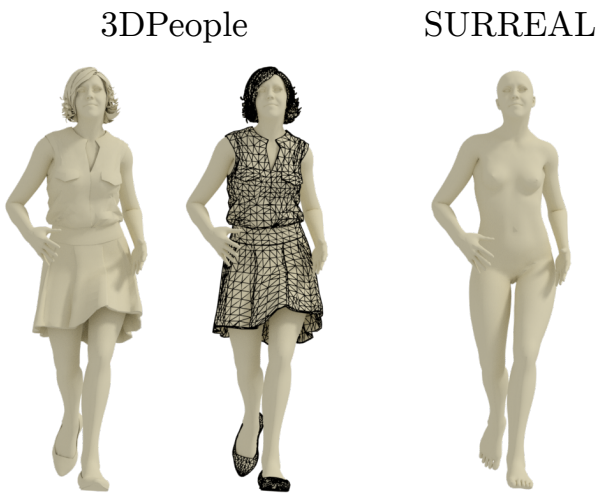


Figure 3.21: **Comparison with previous large-scale datasets.** The only publicly available large-scale dataset of 3D bodies is SURREAL [3]. However, it does not contain true geometry for the clothes, and it projects textures into the naked SMPL shape resembling body-painting. To fill this gap we introduce 3DPeople, a large-scale dataset with true geometry for clothes, hair and apparel.

Additionally, the datasets acquired this way [202, 203, 203] are mostly indoors. Even more complex is the task of obtaining 3D body shape, which requires expensive setups with muti-cameras or 3D scanners. Marcard *et al.* [220] proposed solution based on IMUs and a moving camera but it still does not provide perfect ground-truth annotation. To overcome this situation, datasets with synthetic but photo-realistic images have emerged as a tool to generate massive amounts of training data. SURREAL [3] is the only large scale and more complete dataset so far, with more than 6M frames generated by projecting synthetic textures of clothes onto random SMPL body shapes. The dataset is further annotated with body masks, optical flow and depth. However, since clothes are projected onto the naked SMPL shapes just as textures, they cannot be explicitly modeled (see Fig. 3.21).

To fill this gap, we introduce 3DPeople, the first dataset of dressed humans with specific geometry representation for the clothes. The dataset contains 2 Million photorealistic 640×480 images split into 40 male/40 female performing 70 actions. For every subject-action sequence we randomly change the texture of the clothes, the lighting direction and the background, and capture it from 4 camera views. Each frame is annotated with (see Fig. 3.22): 3D textured mesh of the naked and dressed body; 3D skeleton; normals; body parts and cloth segmentation masks; depth map; optical flow; and camera parameters. In the following we describe the generation process:

Body models: We have generated fully textured triangular meshes for 80 human characters

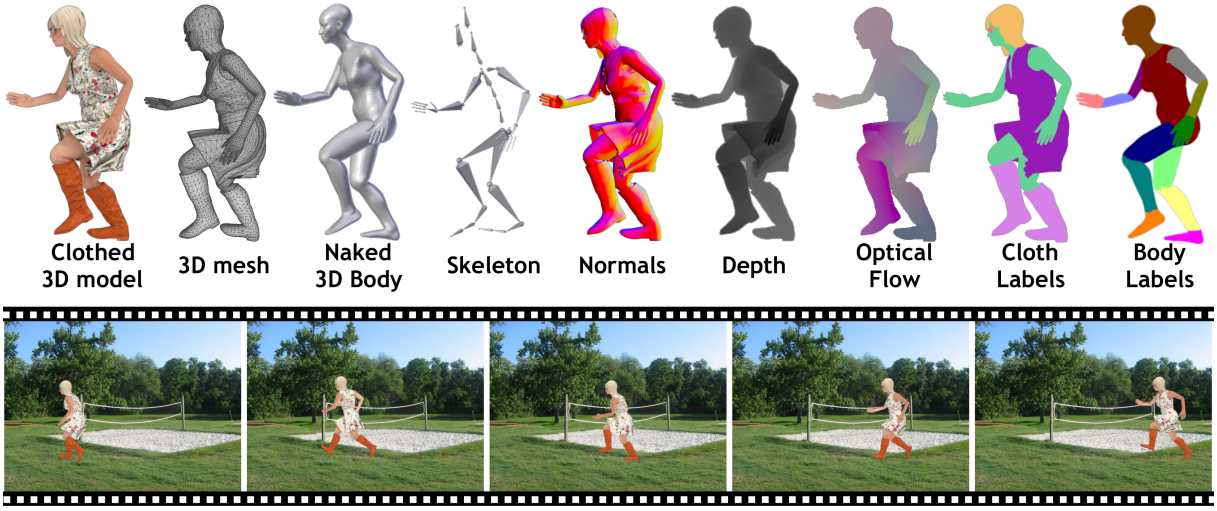


Figure 3.22: **Annotations of the 3DPeople dataset.** For each of the 80 subjects of the dataset, we generate 280 video sequences (70 actions seen from 4 camera views). The bottom of the figure shows 5 sample frames of the *Running* sequence. Every RGB frame is annotated with the information reported in the top of the figure. 3DPeople is the first large-scale dataset with geometric meshes of body and clothes.

using Adobe Fuse [221] and MakeHuman [222]. The distribution of the subjects physical characteristics cover a broad spectrum of body shapes, skin tones and hair geometry (see Fig. 3.23).

Clothing models: Each subject is dressed with a different outfit including a variety of garments, combining tight and loose clothes. Additional apparel like sunglasses, hats and caps are also included. The final rigged meshes of the body and clothes contain approximately 20K vertices.

MoCap sequences: We gather 70 realistic motion sequences from Mixamo [223]. These include human movements with different complexity, from *drinking* and *typing* actions that produce small body motions to actions like *breakdance* or *backflip* that involve very complex patterns. The mean length of the sequences is of 110 frames. While these are relatively short sequences, they have a large expressivity, which we believe make 3DPeople also appropriate for exploring action recognition tasks.

Textures, camera, lights and background: We then use Blender [224] to apply the 70 MoCap animation sequences to each character. Every sequence is rendered from 4 camera views, yielding a total of 22,400 clips. We use a projective camera with a 700 mm focal length and 640×480 pixel resolution. The 4 viewpoints correspond approximately to orthogonal directions aligned with the ground. The distance to the subject changes for every sequence to ensure a full view of the body in all frames. The textures of the clothes are randomly changed for every sequence (see again Fig. 3.20). The illumination is composed of an ambient lighting plus a light

Figure 3.23: **Dataset high variance.** When building the dataset an special effort has been made into covering the broadest spectrum of unbiased body and cloth types. The dataset contains a large variety of shapes, skin tones, hair and beard geometry. Also, each person outfit is unique and complemented with diverse garments such as hats, caps and glasses.

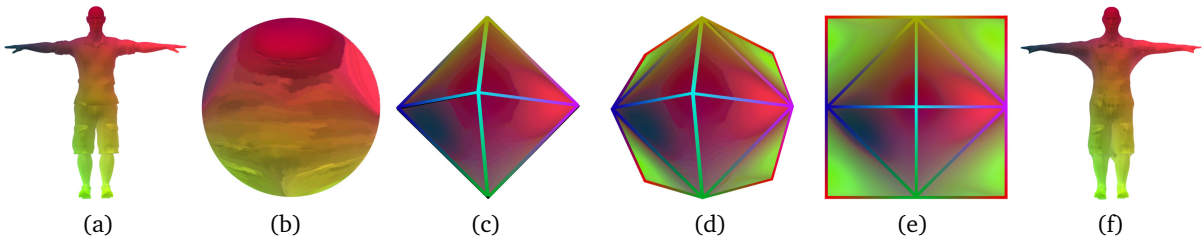


Figure 3.24: **Geometry image representation of the reference mesh**. (a) Reference mesh in a tpose configuration color coded using the xyz position. (b) Spherical parameterization; (c) Octahedral parameterization; (d) Unwarping the octahedron to a planar configuration; (e) Geometry Image, resulting from the projection of the octahedron onto a plane; (f) mesh reconstructed from the Geometry Image. Colored edges in the octahedron and in the Geometry Image represent the symmetry that is later exploited by the mesh regressor Φ.

source at infinite, which direction is changed per sequence. As in [3] we render the person on top of a static background image, randomly taken from the LSUN dataset [225].

Semantic labels: For every rendered image, we provide segmentation labels of the clothes (8 classes) and body (14 parts). Observe in Fig. 3.22-top-right that the former are aligned with the dressed human, while the body parts are aligned with the naked body.

3.6.2 Problem Formulation

Given a single image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$ of a person wearing an arbitrary outfit, we aim at designing a model capable of directly estimating the 3D shape of the clothed body. We represent the body shape through the mesh associated to a Geometry Image with N^2 vertices $\mathbf{X} \in \mathbb{R}^{N \times N \times 3}$ where $\mathbf{x}_i = (x_i, y_i, z_i)$ are the 3D coordinates of the i-th vertex, expressed in the camera coordinates system and centered on the root joint $\mathbf{x}_r$. This representation is a key ingredient of our design, as it maps the 3D mesh to a regular 2D grid structure that preserves the neighborhood relations, fulfilling thus the locality assumption required in CNN architectures. Furthermore, the Geometry Image representation allows uniformly reducing/increasing the mesh resolution by simply uniformly downsampling/upsampling. This will play an important role in our strategy of designing a coarse-to-fine shape estimation approach.

We next describe the two main steps of our pipeline: (i) the process of constructing the Geometry Images, and (ii) the deep generative model we propose for predicting 3D shape.

3.6.3 Geometry Image for Dressed Humans

The deep network we describe later will be trained using pairs $\{\mathbf{I}, \mathbf{X}\}$ of images and their corresponding Geometry Image. For creating the Geometry Images we consider two different cases, one for a reference mesh in a tpose configuration, and another for any other mesh of the dataset.

Geometry Image for a Reference Mesh

One of the subjects of our dataset in a tpose configuration is chosen as a reference mesh. The process for mapping this mesh into a planar regular grid is illustrated in Fig. 3.24. It involves the following steps:

Repairing the mesh. Let $\mathbf{R}^{\text{tpose}} \in \mathbb{R}^{N_R \times 3}$ be the reference mesh with N_R vertices in a tpose configuration (Fig. 3.24-a). We assume this mesh to be a manifold mesh and to be genus-0. Most of the meshes in our dataset, however, do not fulfill these conditions. In order to fix the mesh we follow the heuristic described in [5] which consists of a voxelization, a selection of the largest connected region of the α-shape, and subsequent hole filling using a medial axis approach. We denote by $\tilde{\mathbf{R}}^{\text{tpose}}$ the repaired mesh.

Spherical parameterization. Given the repaired genus-0 mesh $\tilde{\mathbf{R}}^{\text{tpose}}$, we next compute the spherical parameterization $\mathcal{S} : \tilde{\mathbf{R}}^{\text{tpose}} \rightarrow \mathbf{S}$ that maps every vertex of $\tilde{\mathbf{R}}^{\text{tpose}}$ onto the unit sphere $\mathbf{S}$ (Fig. 3.24-b). Details of the algorithm we use are explained below.

Unfolding the sphere. The sphere $\mathbf{S}$ is mapped onto an octahedron and then cut along edges

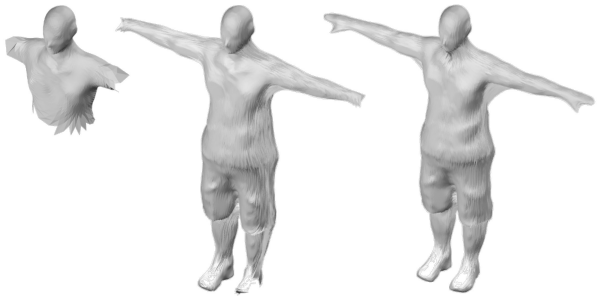


Figure 3.25: **Comparison of spherical mapping methods.** Shape reconstructed from a Geometry Image obtained with three different algorithms. Left: FLASH [4]; Center: [5]; Right: SAPP algorithm we propose. Note that SAPP is the only method that can effectively recover feet and hands.

to output a flat Geometry Image $\mathbf{X}$. Let us formally denote by $\mathcal{U} : \mathbf{S} \rightarrow \mathbf{X}$, and by $\mathcal{G}^R = \mathcal{U} \circ \mathcal{S} : \tilde{\mathbf{R}}^{\text{tpose}} \rightarrow \mathbf{X}$ the mapping from the reference mesh to the Geometry Image. The unfolding process is shown in Fig. 3.24-(c,d,e). Color lines in the Geometry Image correspond to the same edge in the octahedron, and are split after the unfolding operation. We will later enforce this symmetry constraint when predicting Geometry Images.

Spherical Area-Preserving Parameterization

Although there exist several spherical parameterization schemes (*e.g.* [4,5]) we found that they tend to shrink the elongated parts of the full body models such as the arms and legs, making the Geometry Images incomplete (see Fig. 3.25). In this work, we develop a spherical area-preserving parameterization algorithm for genus-0 full body models by combining and extending the FLASH method [4] and the optimal mass transportation method [216]. Our algorithm is particularly advantageous for handling models with elongated parts. The key idea is to begin with an initial parameterization onto a planar triangular domain with a suitable rescaling correcting the size of it. The area distortion of the initial parameterization is then reduced using quasi-conformal composition. Finally, the spherical area-preserving parameterization is produced using optimal mass transportation followed by the inverse stereographic projection.

Geometry Image for Arbitrary Meshes

The approach for creating the Geometry Image described in the previous subsection is quite computationally demanding (up to 15 minutes for complex meshes). To compute the Geometry Image for several thousand training meshes we have devised an alternative approach. Let $\mathbf{Q} \in \mathbb{R}^{N_Q \times 3}$ be the mesh of any subject of the dataset under an arbitrary pose (Fig. 3.26-a), and let

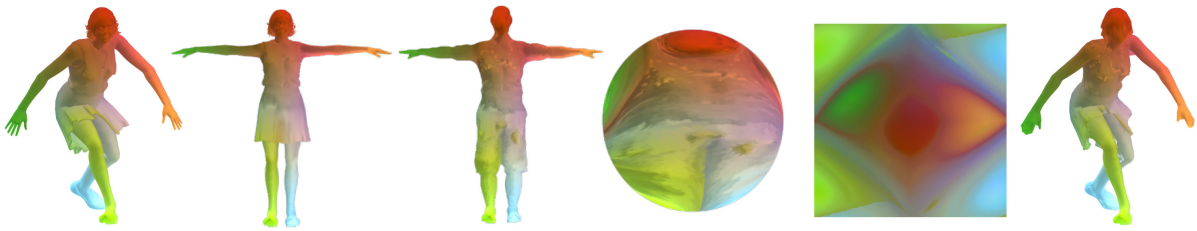


Figure 3.26: **Geometry image estimation for an arbitrary mesh**. (a) Input mesh $\mathbf{Q}$ in an arbitrary pose color coded using the xyz position of the vertices; (b) Same mesh in a tpose configuration ($\mathbf{Q}^{\text{tpose}}$). The color of the mesh is mapped from $\mathbf{Q}$; (c) Reference tpose $\mathbf{R}^{\text{tpose}}$. The colors again correspond from those transferred from $\mathbf{Q}$ through the non-rigid map between $\mathbf{Q}^{\text{tpose}}$ and $\mathbf{R}^{\text{tpose}}$; (d) Spherical mapping of $\mathbf{Q}$; (e) Geometry Image of $\mathbf{Q}$; (f) Mesh reconstructed from the Geometry Image. Note that while being computed through a non-rigid mapping between the two reference poses, the recovered shape is a very good approximation of the input mesh $\mathbf{Q}$.

$\mathbf{Q}^{\text{tpose}} \in \mathbb{R}^{N_Q \times 3}$ be its tpose configuration (Fig. 3.26-b). We assume there is a 1-to-1 vertex correspondence between both meshes, that is, $\exists\ \mathcal{I} : \mathbf{Q} \rightarrow \mathbf{Q}^{\text{tpose}}$ where $\mathcal{I}$ is a known bijective function[5]. We then compute dense correspondences between $\mathbf{Q}^{\text{tpose}}$ and the reference tpose $\tilde{\mathbf{R}}^{\text{tpose}}$, using a Non-Rigid Iterative Closest Point (NR-ICP) algorithm [226]. We denote this mapping as $\mathcal{N} : \mathbf{Q}^{\text{tpose}} \rightarrow \tilde{\mathbf{R}}^{\text{tpose}}$ (see Fig. 3.26-c). We can then finally compute the Geometry Image for the input mesh $\mathbf{Q}$ by concatenating mappings:

$$\mathcal{G}^Q = \mathcal{G}^R \circ \mathcal{N} \circ \mathcal{I} : \mathbf{Q} \rightarrow \mathbf{X} \tag{3.27}$$

where $\mathcal{G}^R$ is the mapping from the reference mesh to the Geometry Image domain estimated above. It is worth pointing that the NR-ICP between the pairs of tposes is also highly computationally demanding, but it only needs to be computed once per every subject of the dataset. Once this is done, the Geometry Image for a new input mesh $\mathbf{Q}$ can be created in a few seconds.

An important consequence of this procedure is that all Geometry Images of the dataset will be semantically aligned, that is, every uv entry in $\mathbf{X}$ will correspond to (approximately) the same semantic part of the model. This will significantly alleviate the learning task of the deep network.

[5]This is guaranteed in our dataset, with all meshes of the same subject having the same number of vertices.

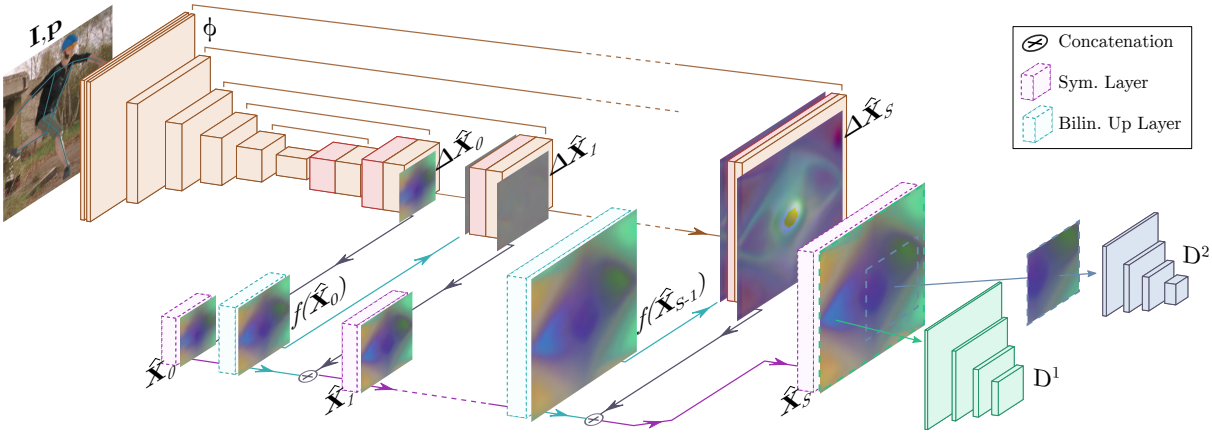


Figure 3.27: **GimNet overview**. The proposed architecture consists of two main blocks: a multiscale Geometry Image regressor Φ and a multiscale discriminator D to evaluate the local and global consistency of the estimated meshes.

3.6.4 GimNet

We next introduce *GimNet*, our deep generative network to estimate Geometry Images (and thus 3D shape) of dressed humans from a single image. An overview of the model is shown in Fig. 3.27. Given the input image, we first extract the 2D joint locations $\mathbf{p}$ represented as heatmaps [23, 194], which are then fed into a mesh regressor $\Phi(\mathbf{I}, \mathbf{p})$ trained to reconstruct the shape $\hat{\mathbf{X}}$ of the person in $\mathbf{I}$ employing a Geometry Image based representation. To enforce the reconstruction to lie on the manifold of anthropomorphic shapes, an adversarial scheme with two discriminators D^1 and D^2 is applied.

Mesh Regressor. Given the input image $\mathbf{I}$ and the estimated 2D body joints $\mathbf{p}$, the mesh regressor Φ aims to predict the Geometry Image $\mathbf{X}$, *i.e.*we seek to estimate the mapping $\mathcal{M}$: $\mathbf{I}, \mathbf{p} \rightarrow \mathbf{X}$. Instead of directly learning the complex mapping $\mathcal{M}$, we break the process into a sequence of more manageable steps. Φ initially estimates a low-resolution mesh, and then progressively increases its resolution (see Fig. 3.27). This coarse-to-fine approach allows the regressor to first focus on the basic shape configuration and then shift attention to finer details, while also providing more stability compared to a network that learns the direct mapping.

As shown in Fig. 3.24-e, the Geometry Images have symmetry properties derived from unfolding the octahedron into a square, specifically, each side of the Geometry Image is symmetric with respect to its midpoint. We force this property using a differentiable layer that linearly operates over the edges of the estimated Geometry Images.

Multi-Scale Discriminator. Evaluating high-resolution meshes poses a significant challenge for a discriminator, as it needs to simultaneously guarantee local and global mesh consistency on very high dimensional data. We therefore use two discriminators with the same architecture, but that operate in different Geometry Image scales: (i) a discriminator with a large receptive field that evaluates the shape coherence as a whole; and (ii) a local discriminator that focuses on small patches and enforces the local consistency of the surface triangle faces.

3.6.5 Learning the Model

3D Reconstruction Error. We first define a supervised multi-level L1 loss for 3D reconstruction $\mathcal{L}_{\text{R}}$ as:

$$\mathcal{L}_{\text{R}} = \mathbb{E}_{\mathbf{X}\sim\mathbb{P}_r,\hat{\mathbf{X}}\sim\mathbb{P}_g} \frac{1}{S}\sum_{s=1}^{S} \lambda_s \left\| \mathbf{X}_s - \hat{\mathbf{X}}_s \right\|_1 , \tag{3.28}$$

being $\mathbb{P}_r$ and $\mathbb{P}_g$ the real and generated data distribution of clothed human Geometry Images respectively, S the number of scales, $\mathbf{X}_s$ the ground-truth reconstruction at scale s and $\hat{\mathbf{X}}_s = \Phi_s(\mathbf{I})$ the estimated reconstruction. The error at each scale is weighted by $\lambda_s = \frac{1}{r}$ where r is the ratio between $\hat{\mathbf{X}}_S$ and $\hat{\mathbf{X}}_s$ sizes. During initial experimentation L1 loss reported better reconstructions than mean squared error.

2D Projection Error. To encourage the mesh to correctly project onto the input image we penalize, at every scale s, its projection error $\mathcal{L}_{\text{P}}$ computed as:

$$\mathcal{L}_{\text{P}} = \mathbb{E}_{\mathbf{X}\sim\mathbb{P}_r,\hat{\mathbf{X}}\sim\mathbb{P}_g} \frac{1}{S}\sum_{s=1}^{S} \lambda_s \left\| \mathcal{P}(\mathbf{X}_s) - \mathcal{P}(\hat{\mathbf{X}}_s) \right\|_1 ,$$

where $\mathcal{P}$ is the differentiable projection equation and λ_s is calculated as above.

Adversarial Loss. In order to further enforce the mesh regressor Φ to generate anthropomorphic shapes we perform a min-max strategy game [40] between the regressor and two discriminators operating at different scales. It is well-known that non-overlapping support between the true data distribution and model distributions can cause severe training instabilities. As proven by [227, 228], this can be addressed by penalizing the discriminator when deviating from the Nash-equilibrium, ensuring that its gradients are non-zero orthogonal to the data

manifold. Formally, being D^k the kth discriminator, the $\mathcal{L}_{\text{adv}}$ loss is defined as:

$$\sum_{k=1}^{K}\Big[\mathbb{E}_{\hat{\mathbf{X}}\sim\mathbb{P}_g}[\log(1-D^k(\hat{\mathbf{X}}_S))]+\mathbb{E}_{\mathbf{X}\sim\mathbb{P}_r}\Big[\log(D^k(\mathbf{X}_S))\Big] + \frac{\lambda_{\text{dgp}}}{2}\mathbb{E}_{\mathbf{X}\sim\mathbb{P}_r}(\|\nabla D^k(\mathbf{X}_S)\|_1^2\Big], \tag{3.29}$$

where $K=2$ and λ_{dgp} is a penalty regularization for discriminator gradients, only considered on the true data distribution.

Feature Matching Loss. To improve training stabilization we penalize higher level features on the discriminators [229]. Similar to a perception loss, the estimated Geometry Image is compared with the ground truth at multiple feature levels of the discriminators. Being D_l^k the l-th layer of the k-th discriminator, $\mathcal{L}_{\text{F}}$ is defined as:

$$\mathbb{E}_{\mathbf{X}\sim\mathbb{P}_r,\hat{\mathbf{X}}\sim\mathbb{P}_g}\sum_{k=1}^{K}\sum_{l=1}^{L}\frac{1}{N_l^k}\left\|D_l^k(\mathbf{X}_S)-D_l^k(\hat{\mathbf{X}}_S)\right\|_1, \tag{3.30}$$

where N_l^k is a weight regularizer denoting the number of elements in the l-th layer of the kth discriminator.

Total Loss. Finally, we to solve the min-max problem:

$$\Phi^\star = \arg\min_{\Phi}\max_{D}\mathcal{L}_{\text{adv}}+\lambda_{\text{R}}\mathcal{L}_{\text{R}}+\lambda_{\text{P}}\mathcal{L}_{\text{P}}+\lambda_{\text{F}}\mathcal{L}_{\text{F}} \tag{3.31}$$

where λ_{R}, λ_{P} and λ_{F} are the hyper-parameters that control the relative importance of every loss term.

Implementation Details

For the mesh regressor Φ we build upon the U-Net architecture [230] consisting on an encoder-decoder structure with skip connections between features at the same resolution extended to estimate Geometry Images at multiple scales. The encoder transforms the original image into a 6 levels pyramid of multi-scale high-dimensional features $\mathbf{H}_s$, each obtained by an average pooling with stride 2 to down-sample the spatial dimension followed by two convolution blocks. On the opposite site, the decoder builds upon a cascade of Geometry Image inference blocks that estimate a coarse-to-fine mesh at multiple resolutions. The Geometry Image at each level is estimated as $\hat{\mathbf{X}}_s = f(\hat{\mathbf{X}}_{s-1})+\Delta\hat{\mathbf{X}}_s$ being $f(\hat{\mathbf{X}}_{s-1})$ the bilinearly upsampled Geometry Image of the previous level. By doing this our network only needs to infer a residual delta $\Delta\hat{\mathbf{X}}_s$

	mean	median	std	time (sec)
Choi 2015	333.16	331.68	47.66	3.66
Sinha 2016	24.90	12.86	36.76	268.94
Ours SaP	14.02	10.30	14.59	100.68
Ours SaP + NR-ICP	10.37	9.98	1.86	1.00
NR-ICP (lower bound)	4.89	4.67	0.83	-

Table 3.7: **GIM accuracy.** The errors are in mm, using the defined distance metric. We also report the time required to compute the GIM for a single mesh. These results are obtained over the t-pose configurations of the 80 models of our dataset.

over a lower dimensional mesh instead of the complete mesh. $\Delta\hat{\mathbf{X}}_s$ is estimated by applying a convolutional block with a Tanh$(\cdot)$ activation over the concatenation of bilinearly upsampled features of the previous level $f(\mathbf{F}_{s-1})$, the estimated mesh in a previous resolution $f(\hat{\mathbf{X}}_{s-1})$, and the skip connection $\mathbf{H}_s$. For $s = 0$, the lowest dimensional mesh, $\hat{\mathbf{X}}_0 = \Delta\hat{\mathbf{X}}_0$. In the finest scale S, two refinement convolutions produce the final estimate $\hat{\mathbf{X}}_S$. All convolution blocks, except for the first two, consist of a convolution layer with stride and padding 1 and 3×3 kernels followed by instance normalization and a Leaky ReLU activation with $\alpha = 0.2$. The first two blocks in the first encoder level use 7×7 kernels to capture long range dependencies of the shape at high resolution.

Both discriminator networks operate at different mesh resolutions [229] but have the same PatchGan [52] architecture mapping from the Geometry Image $\mathbf{X}$ to a matrix $\mathbf{Y} \in \mathbb{R}^{H/8 \times W/8}$, where $\mathbf{Y}[i, j]$ represents the probability of the patch ij to be close to a real Geometry Image distribution. The global discriminator evaluates the final mesh resolution at scale S and the local discriminator the down-sampled mesh at scale $S - 1$. Each discriminator is composed by 3 down-convolution blocks, each containing 4×4 convolutions with stride 2 followed by instance normalization and Leaky ReLU with $\alpha = 0.2$; and a final convolution to produce the 1 dimensional estimate.

The model is trained with 170,000 synthetic images of cropped clothed people resized to 128×128 pixels and Geometry Images of $128 \times 128 \times 3$ (meshes with 16,384 vertices) during 60 epochs and $S = 4$. As for the optimizer, we use Adam [38] with learning rate of $2e-4$, beta1 0.5, beta2 0.999 and batch size 110. Every 40 epochs we decay the learning rate by a factor of 0.5. The weight coefficients for the loss terms are set to $\lambda_{\text{R}} = 20$, $\lambda_{\text{P}} = 0.1$, $\lambda_{\text{F}} = 10$ and $\lambda_{\text{dgp}} = 0.01$.

3.6.6 Experimental Evaluation

We next present quantitative and qualitative results of the proposed Geometry Image mapping and our model ability to perform 3D reconstruction on synthetic images of our dataset and on

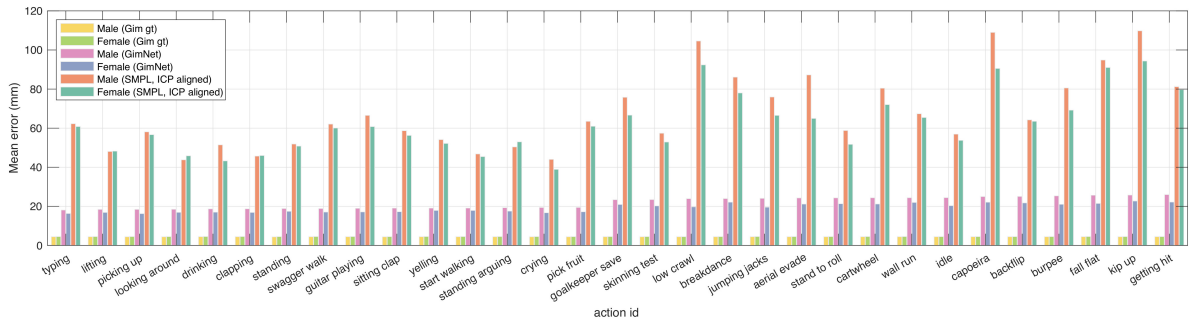


Figure 3.28: **Mean error distance on the test set.** We plot the results for the 15 worst and 15 best actions. Besides the results of GimNet, we report the results obtained by the ground truth GIM (recall that it is an approximation of the actual ground truth mesh). We also display the results obtained by the recent parametric approach of [6]. The results of this method, however are merely indicative, as we did not retrain the network with our dataset.

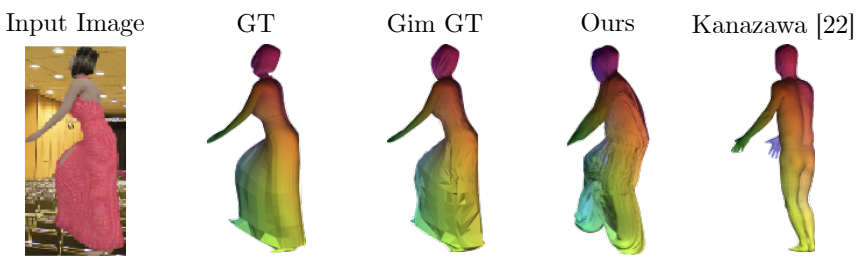


Figure 3.29: **Qualitative comparison with baseline**. Given the input image on the left we plot the results estimated by our method and [6] (last to columns). We also present the ground truth mesh and GIM (recall that it is an approximation of the ground truth mesh).

images in the wild.

GIM Generation - Mapping Baseline Comparison

We first provide a detailed analysis of the proposed spherical area-preserving parametrization. We compare our approach against the baselines Choi *et al.* [4] and Sinha *et al.* [5]. In Table 3.7 we report the reconstruction error of the GIMs generated by these two methods, *Ours SaP*: the Spherical area Preserving strategy described in Sec. 3.6.3; and *Ours SaP + NR-ICP*: the combined SaP and NR-ICP mapping to a reference shape. Additionally, we provide the error of the NR-ICP mapping, between each input mesh and a reference shape being this the lower-bound error we can obtain using *Ours SaP + NR-ICP*. We found that [4, 5] tend to shrink the elongated parts of the full body models such as the arms and legs, making the Geometry Images incomplete (see Fig. 3.25). In contrast, our approach is particularly advantageous for handling these cases being consistently more accurate and computationaly efficient than the rest of baselines.

Noise Uniform Range (in pixels)	{0}	[-2, +2]	[-4, +4]	[-8, +8]	[-10, +10]
Mean Error	23.29	24.93	30.19	49.73	61.28

Table 3.8: **Model Sensitivity to noise in the 2D joints.** Reconstruction error in mm, when injecting uniform noise in the ranges $\pm\{0, 2, 4, 8, 10\}$ pixels.

3D Reconstruction - Synthetic Results

We evaluate our approach on 25,000 test images randomly chosen for 8 subjects (4 male/ 4 female) of the test split. For each test sample we feed GimNet with the RGB image and the ground truth 2D pose, corrupted by Gaussian noise with 2 pixel std. For a given test sample, let $\hat{\mathbf{Y}}$ be the $N^2 \times 3$ estimated mesh, resulting from a direct reshaping of its estimated Geometry Image $\hat{\mathbf{X}}$. Also, let $\mathbf{Y}$ be the ground truth mesh, which does not need to have neither the same number of vertices as $\tilde{\mathbf{Y}}$, nor necessarily the same topology. Since there is no direct 1-to-1 mapping between the vertices of the two meshes we propose using the chamfer metric:

$$\text{dist}(\hat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{2}(\text{KNN}(\hat{\mathbf{Y}} \rightarrow \mathbf{Y}) + \text{KNN}(\mathbf{Y} \rightarrow \hat{\mathbf{Y}})) \tag{3.32}$$

where $\text{KNN}(\hat{\mathbf{Y}} \rightarrow \mathbf{Y})$ represents the average Euclidean distance for all vertices of $\hat{\mathbf{Y}}$ to their nearest neighbor in $\mathbf{Y}$. Note that $\text{KNN}(\cdot, \cdot)$ is not a true distance measure because it is not symmetric. This is why we compute it bidirectionally.

The quantitative results are summarized in Fig. 3.28. We report the average error (in mm) of GimNet for 30 actions (the 15 with the highest and lowest error). Note that the error of GimNet is bounded between 15 and 35mm. Recall, however, that we do not consider outlier 2D detections in our experiments, but just 2D noise. We also evaluate the error of the ground truth Geometry Image, as it is an approximation of the actual ground truth mesh. This error is below 5mm, indicating that the Geometry Image representation does indeed capture very accurately the true shape. We also provide the error of the recent parametric approach of [6], that fits SMPL parameters to the input images. Nevertheless, these results are just indicative, and cannot be directly compared with our approach, as we did not retrain [6]. We add them here just to demonstrate the challenge posed by the new 3DPeople dataset. Indeed, the distance error in [6] was computed after performing a rigid-icp of the estimated mesh with the ground truth mesh (there was no need of this for GimNet).

Fig. 3.29 shows a representative example of the meshes used in the quantitative comparison. As a common denominator across all test set, [6] struggles with extremely occluded limbs but most importantly fails to capture the cloth due to its underlying SMPL representation.

Figure 3.30: **Qualitative results.** For the synthetic images we plot our estimated results and the shape reconstructed directly from the ground truth Geometry Image. In all cases we show two different views. The color of the meshes encodes the xyz vertex position.

Finally, we evaluate the method sensitivity to nosy skeleton inputs. Table 3.8 reports the average 3D reconstruction error (in mm) of GimNet for different levels of 2D noise of the input skeleton joins. Note that the error is below reasonable bounds for noise levels up to 4 pixels. Indicating that GimNet does indeed relay on the skeleton as a first approximation constrain while still being robust to non perfect skeleton estimations up to 4 pixels for images of 128×128.

3D Reconstruction - Qualitative Results

We finally show in Fig. 3.30 qualitative results on synthetic images from 3DPeople and real fashion images downloaded from Internet. Remarkably, note how our approach is able to reconstruct long dresses (top row images), known to be a major challenge [159]. Note also that some of the reconstructed meshes have spikes. This is one of the limitations of the non-parametric models, that the reconstructions tend to be less smooth than when using parametric fittings. However, non-parametric models have also the advantage that, if properly trained, can span a much larger configuration space.

3.7 Summary

In this chapter we have presented four novel methods that define a pipeline for reconstructing a scene and clothed people in it while estimating the camera position.

For scene reconstruction and camera estimation we have proposed PL-SLAM, an approach to visual SLAM that allows to simultaneously process points and lines and tackle situations where point-only based methods are prone to fail, like poorly textured scenes or motion blurred images where feature points are vanished out. We built upon the architecture of the state-of-the-art ORB-SLAM and modify its original pipeline to operate with line features without significantly compromising its efficiency. We have also presented a novel line-based map initialization approach, which estimates camera pose and 3D map from 5 line correspondences in three consecutive images. This approach holds on the assumption of constant and small inter-frame rotation in these three images. In the results section we show that this indeed is a good approximation for many situations. Additionally, we evaluated the full pipeline on a real robot for the TUM RGB-D benchmark and showed consistent improvement w.r.t. current competing methods.

To segment and track the objects in the scene to be later reconstructed we have presented FaSTGAN, the first real-time approach for semisupervised video object segmentation running at 32 FPS and yielding high-quality segmentation masks. FaSTGAN's accuracy is on a par with previous state of the art optimized for speed but it runs at a much higher frame rate. To achieve this, we have designed a novel GAN architecture made of a relatively small regressor and two critics that enforce spatio-temporal consistency over finite temporal windows during training. At test time, the critics are removed, leading to a simple but robust regressor that does not require fine-tuning nor post-processing operations when applied to new sequences with unseen objects.

We then have proposed DeformNet, the first deep network for 3D shape estimation of a non-rigid surface from a single image. For this purpose we have designed an architecture that can be trained in an end-to-end manner, but that internally splits the problem in three stages: 2D detection, depth estimation and shape inference. The three stages are intimately connected and are executed by ensuring the satisfaction of geometric constraints such as correct 3D-to-2D reprojection and 3D-to-3D alignment between the estimated and the ground truth shapes. In order to train this network, we have rendered a large synthetic dataset of shapes under different levels of deformation, varying textures, material properties and illumination conditions. We have shown this network to outperform existing analytical solutions while being much more efficient and allowing to tackle situations with large amounts of occlusion and very poorly textured surfaces.

Finally, we have tackled the more complex task of 3D reconstruction of clothed people. To do so, we have made three contributions: (i) we have presented the first large-scale dataset of 3D humans in action in which cloth geometry is explicitly modelled; (ii) we have proposed a new algorithm to perform spherical parameterizations of elongated body parts, to later model rigged meshes of human bodies as geometry images; and (iii) we have introduced an end-to-end network to estimate human body and clothing shape from single images, without relying on parametric models.

4

Image & Video Person Synthesis

4.1 Introduction

In this chapter we present three novel generative methods to model the nonrigid articulated human body. All proposed methods are self-supervised, they do not required specific dataset with ground truth labels of the specific task. The experimentation results demonstrate that these methods are capable of generating novel photo-realistic views of a person, face expressions and cloth in the wild.

Addressing these tasks is an extremely complex endeavor. Nevertheless, GANs have shown impressive results in rendering new realistic images, *e.g.*., faces [9,231], indoor scenes [232] and clothes [233], by directly learning a generative model from data. They have been used for the particular problem of person image generation and edition from single-view images [14, 234]. While the results of these approaches are very promising, they suffer from the same fundamental limitation in that are methods trained in a fully supervised manner, that is, they need to be trained with pairs of input-output images. This requires from specific datasets. Tackling the problem in an unsupervised manner, one could leverage to an unlimited amount of images and use other datasets for which no multi-view images of people are available.

First, we present a novel approach for synthesizing photorealistic images of people in arbitrary poses using generative adversarial learning (Sec. 4.3). Given an input image of a person and a desired pose represented by a 2D skeleton, our model renders the image of the same person under the new pose, synthesizing novel views of the parts visible in the input image and hallucinating those that are not seen. We tackle this challenging scenario by splitting the problem into two principal subtasks: (i) We consider a pose conditioned bidirectional generator that maps back the initially rendered image to the original pose, hence being directly comparable to the input image without the need to resort to any training image. (ii) We devise a novel loss function that incorporates content and style terms, and aims at producing images

of high perceptual quality. Extensive experiments conducted on the DeepFashion dataset [7] demonstrate that the images rendered by our model are very close in appearance to those obtained by fully supervised approaches.

Second, we present GANimation, a novel GAN conditioning scheme for facial expression editing based on Action Units (AUs) annotations, which describes in a continuous manifold the anatomical facial movements defining human expressions (Sec. 4.4). GANimation allows controlling the magnitude of activation of each AU and combine several of them. Additionally, we propose a fully unsupervised strategy to train the model, that only requires images annotated with their activated AUs, and exploit attention mechanisms that make our network robust to changing backgrounds and lighting conditions. Extensive evaluation shows that our approach goes beyond competing conditional generators both in the capability to synthesize a much wider range of expressions ruled by anatomically feasible muscle movements, as in the capacity of dealing with images in the wild.

Finally, we tackle the virtual dressing problem as a cloth transfer one, in which given a single image of a subject, we transfer his/her clothes to one or more target subjects, either in still images or video sequences (Sec. 4.5). The core of the presented approach builds upon a time-consistent GAN with a memory module that stores and progressively refines a texture map representation of the clothes by hallucinating parts which were not initially visible. Again, this model can be trained in an unsupervised manner, that is, it does not require being trained with pairs of images of the same subject wearing different clothes. A thorough evaluation shows that our approach captures very well the clothing appearance under varying body poses, complex lighting and changing backgrounds, and provides temporally consistent mappings in long video sequences.

4.2 Related Work

Rendering a person in an arbitrary pose, expression and cloth from a single image is a severely ill-posed problem as there are many cloth and body shape ambiguities caused by the new camera view and the changing body pose, as well as large areas of missing data due to body self-occlusions. Solving such a rendering problem requires introducing several sources of prior knowledge including, among others, the body shape, kinematic constraints, hair dynamics, cloth texture, reflectance models and fashion patterns.

Initial solutions to tackle this problem first built a 3D model of the object and then synthesized the target images under the desired views [235–237]. These methods, however, were constrained to rigid objects defined by either computer-aided designed models or relatively simple geometric primitives. Other strategies constrain the set of possible configurations to

those that can be generated using low-rank 3D Morphable Models (3DMMs). Early approaches along this line [134,238] generated novel expressions by adjusting the initially estimated 3DMM parameters of a registered face. While simple, this strategy produced strong image artifacts and could not convey shading and illumination effects.

More recently, with the advent of Deep Learning and GANs [40], there has been a growing interest in learning generative image models from data. Particularly interesting for this section are those approaches that incorporate conditions to train GANs and constrain the generation process. Several conditions have been explored so far, such as discrete labels [47, 48], and text [45]. Images have also been used as a condition, for instance in the problem of image-to-image translation [52], for future frame prediction [50], image inpainting [51] and face alignment [239]. Very recently [233] used both textual descriptions and images as a condition to generate new clothing outfits.

Similar to the methods proposed in this thesis, several works have tackled the problem of unpaired training data. First attempts [240] relied on Markov random field priors for Bayesian based generation models, using images from the marginal distributions in individual domains. Others explored enhancing GANs with VAE strategies [240, 241]. Later, several works [51, 242] have exploited the idea of driving the system to produce mappings transforming the style without altering the original input image content. Our approach is more related to those works exploiting cycle consistency to preserve key attributes between the input and the mapped image, such as CycleGAN [243], DiscoGAN [244] and StarGAN [11].

We next elaborate on the state-of-the-art of all necessary aspects required to synthesize a person in an arbitrary pose, expression and cloth from a single image:

Person Pose Retargeting. The works that are most related to ours are [14, 234]. They both propose GAN models for the muti-view person image generation problem. However, the two approaches use ground-truth supervision during train, *i.e.*, pairs of images of the same person in two different poses dressed the same. Tackling the problem in a fully unsupervised manner, as we do in Sec. 4.3 chapter, becomes a much harder task that requires more elaborate network designs, specially when estimating the loss of the rendered images.

Face Image Editing. Face generation and editing is a well-studied topic in computer vision and generative models. Early approaches addressed the problem using mass-and-spring models to physically approximate skin and muscle movement [245]. The problem with this approach is that it is difficult to generate natural looking facial expressions able to capture subtle skin movements with simple spring models. [238] relied on 2D and 3D morphings but produced strong artifacts around the region boundaries and was not able to model illumination changes.

More recent works [8, 11, 48] train complex convolutional networks able to work with images in the wild. However, these approaches have been conditioned on discrete emotion categories (*e.g.*, happy, neutral, and sad). Instead, in Sec. 4.4 we resume the idea of modeling skin and muscles, but we integrate it in modern Deep Learning machinery. More specifically, we learn a GAN model conditioned on a continuous embedding of muscle deformations, allowing to generate a large range of anatomically possible face expressions as well as smooth facial movement transitions in video sequences.

Cloth Transfer. 3D cloth reconstruction [23, 24, 105, 198] is an open research problem which is receiving increasing attention and that it can be an essential tool when building virtual try-on systems. One of the first works in going a step further and exploring ways to generate novel images of dressed people is [246], that proposed a system for animating realistic clothing on synthetic 3D bodies under arbitrary poses. In [247] a multi-part 3D model of clothed-bodies is used to reconstruct and retarget cloth to new body shapes. Later, Deep Learning entered into action with several works on this topic: [25] proposed an unsupervised approach for cloth synthesis in arbitrary poses, [248] trained a network to render a cloth item in a person given a template image and [249] exploited a person 3D model [138] to change his/her pose while realistically maintaining the cloth appearance. None of the previous methods could handle images with cluttered background. This was first tackled by [250], that extended DensePose [251] for the task of dense texture transfer. This work, however, fully maps the source appearance and body of the source person onto the target, losing his/her identity.

Background Inpainting. Early approaches for image completion were based on diffusion-based formulations that filled image holes by propagating neighbour pixels [252, 253]. These methods, however, could not handle large holes, and were later extended with patch based strategies [254–256] which filled incomplete regions using patches from the original image. While effective, patch-based methods tend to be computationally demanding and cannot handle non-stationary textures. Recently, DL techniques have surpassed these early approaches. For instance, [257] proposed a dual discriminator loss to assess the image consistency, both locally and globally. Yu *et al.* [258] extended this work by embedding an attention mechanism into a post-processing refinement network. [259] introduced partial convolutions to deal with irregular holes. In Sec. 4.5 we adopt these ideas to fill regions of the image that remain incomplete after an initial mapping.

Memory based Generative Models. When generating video it is necessary to ensure temporal consistency along the regressed frames. Several works have explored the idea of extending DL

networks with memory, in order to backup long-term information that will be later queried for future inferences [260–262]. The main advantage of this strategy w.r.t. to other long-term state-based methods is its ability to robustly secure memory data across long sequences. This idea has been also incorporated into generative models for better data modelling. Arici and Celikyilmaz [263] proposed a Restricted Boltzmann Machine associative memory network extending the GANs framework to transfer features distribution discovered by the discriminator to the generator. Li *et al.* [264] added an external memory with a differentiable controller to capture local texture details that are commonly lost in the bottom-up abstraction process. Kim *et al.* [265] addressed the problem of structural discontinuity and discriminator forgetting when dealing with highly multi-modal latent spaces. In Sec. 4.5 we extend the use of the memory to maintaining long-term texture information for long video sequences generation.

4.3 Unsupervised Person Image Synthesis

In this first section we propose a fully unsupervised GAN framework that, given a photo of a person, automatically generates images of that person under new camera views and distinct body postures. The generative model we build is able to synthesize novel views of the body parts and clothes that are visible in the original image and also hallucinating those that are not seen. As shown in Fig. 4.1, the generated images retain the body shape, and the new textures are consistent with the original image, even when input and desired poses are radically different. In order to learn this model using unlabeled data (*i.e.*, our training data consists of single images of people plus the input and desired poses), we propose a GAN architecture that combines ingredients of the pose conditional adversarial networks [266], Cycle-GANs [243] and the loss functions used in image style transfer that aim at producing new images of high perceptual quality [267].

More specifically, to circumvent the need for pairs of training images of the same person under different poses, we split the problem in two main stages. First, we consider a pose conditioned bidirectional adversarial architecture which, given a single training photo, initially renders a new image under the desired pose. This synthesized image is then rendered-back to the original pose, hence being directly comparable to the input image. Second, in order to assess the quality of the rendered images we devise a novel loss function computed over the 3-tuple of images –original, rendered in the desired pose, and back-rendered to the original pose– that incorporates content and style terms. This function is conditioned on the pose parameters and enforces the rendered image to retain the global semantic content of the original image as well as its style at the joints location.

Extensive evaluation on the DeepFashion dataset [7] using unlabeled data shows very promis-

Figure 4.1: **Model overview.** Given an original image of a person (left) and a desired body pose defined by a 2D skeleton (bottom-row), our model generates new photo-realistic images of the person under that pose (top-row). Our main contribution is to train such generative model with unlabeled data.

ing results, even comparable with recent fully supervised approaches [14, 234].

We have presented a novel approach for generating new images of a person under arbitrary poses using a GAN model that can be trained in a fully unsupervised manner. This advances state-of-the-art, which so far, had only addressed the problem using supervision. To tackle this challenge, we have proposed an new framework that circumvents the need of training data by optimizing a loss function that only depends on the input image and the rendered one, and aims at retaining the style and semantic content of the original image.

In summary, this section main contributions are: 1) a novel approach for generating new images of a person under arbitrary poses using a generative model that can be trained in a fully unsupervised manner; and 2) a new framework that circumvents the need of training data by optimizing a loss function that only depends on the input image and the rendered one, and aims at retaining the style and semantic content of the original image.

4.3.1 Problem Formulation

Given a single-view image of a person, our goal is to train a GAN model in an unsupervised manner, allowing to generate photo-realistic pose transformations of the input image while retaining the person identity and clothes appearance. Formally, we seek to learn the mapping $(\mathbf{I}_{p_o}, \mathbf{p}_f) \rightarrow \mathbf{I}_{p_f}$ between an image $\mathbf{I}_{p_o} \in \mathbb{R}^{3\times H\times W}$ of a person with pose $\mathbf{p}_o$ and the image $\mathbf{I}_{p_f} \in \mathbb{R}^{3\times H\times W}$ of the same person with the desired position $\mathbf{p}_f$. Poses are represented by 2D skeletons with $N = 18$ joints $\mathbf{p} = (\mathbf{u}_1, \ldots, \mathbf{u}_N)$, where $\mathbf{u}_i = (u_i, v_i)$ is the i-th joint pixel

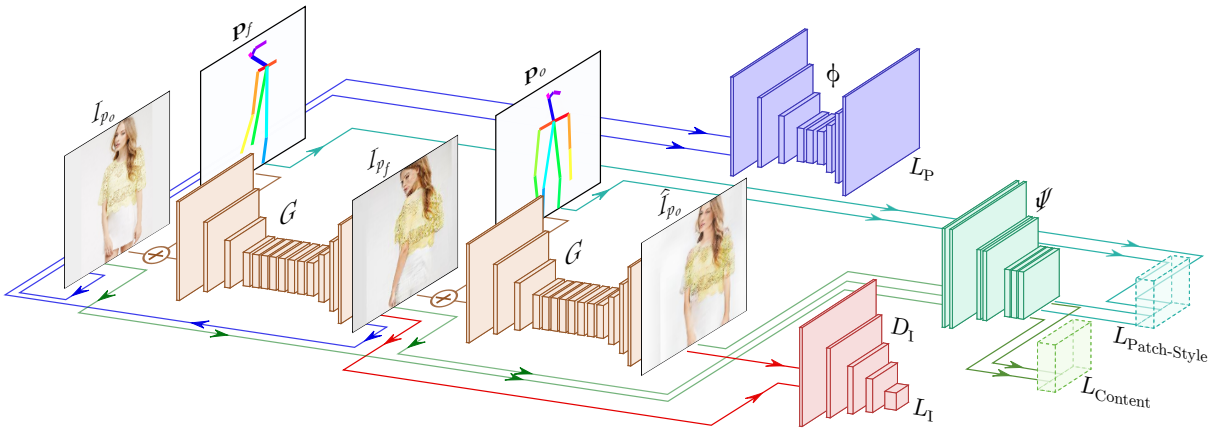


Figure 4.2: **Overview of our unsupervised approach to generate multi-view images of persons.** The proposed architecture consists of four main components: a generator G, a discriminator D, a 2D pose regressor Φ and the pre-trained feature extractor Ψ. Neither ground truth image nor any type of label is considered.

location in the image. The model is trained in an unsupervised manner with training samples $\{\mathbf{I}_{p_o}^{(i)}, \mathbf{p}_o^{(i)}, \mathbf{p}_f^{(i)}\}_{i=1}^{N}$ that *do not* contain the ground-truth output image $\mathbf{I}_{p_f}$.

4.3.2 Unsupervised Generative Model

Fig. 4.2 shows an overview of our model. It is composed of four main modules: (i) A generator $G(\mathbf{I}|\mathbf{p})$ that acts as a differentiable render, mapping one input image of a given person under a specific pose to an output image of the same person under a different pose. Note that G is used twice in our network, first to map the input image $\mathbf{I}_{p_o} \rightarrow \mathbf{I}_{p_f}$ and then to render the latter back to the original pose $\mathbf{I}_{p_f} \rightarrow \hat{\mathbf{I}}_{p_o}$; (ii) A regressor Φ responsible of estimating the 2D joint locations of a given image; (iii) A discriminator $D_{\mathbf{I}}(\mathbf{I})$ that seeks to discriminate between generated and real samples; (iv) A loss function, computed without ground truth, that aims to preserve the person identity. For this purpose, we devise a novel loss function that enforces semantic content similarity of $\mathbf{I}_{p_o}$ and $\hat{\mathbf{I}}_{p_o}$, and style similarity between $\mathbf{I}_{p_o}$ and $\mathbf{I}_{p_f}$.

We next describe in detail each of these components as well as the 2D pose embedding we consider.

Pose Embedding

Drawing inspiration on [194], the 2D location of each skeleton joint $\mathbf{u}_i$ in an image $\mathbf{I} \in \mathbb{R}^{3\times H\times W}$ is represented as a probability density map $\mathbf{B}_i \in \mathbb{R}^{H\times W}$ computed over the entire image domain as:

$$\mathbf{B}_i[u,v] = P(\mathbf{u}_i = (u,v)) \;\; \forall\, (u,v) \in \mathcal{U}, \tag{4.1}$$

being $\mathcal{U}$ the set of all (u, v) pixel locations in the input image $\mathbf{I}$. For each vertex $\mathbf{u}_i$ we introduce a Gaussian peak with variance 0.03 in the position (u_i, v_i) of the belief map $\mathbf{B}_i$. The full person pose $\mathbf{p}$ is represented as the concatenation of all belief maps $\mathbf{p} = (\mathbf{B}_1, \ldots, \mathbf{B}_N) \in \mathbb{R}^{N \times H \times W}$.

Network Architecture

Generator. Given an input image $\mathbf{I}$ of a person, the generator $G(\mathbf{I}|\mathbf{p})$ aims to render a photo-realistic image of that person in a desired pose $\mathbf{p}$. In order to condition the generator with the pose we consider the concatenation $(\mathbf{I}, \mathbf{p}) \in \mathbb{R}^{(N+3) \times H \times W}$ and feed this into a feed forward network that produces an output image with the same dimensions as $\mathbf{I}$. The generator is implemented as the variation of the network from Johnson *et al.* [268] proposed by [243] as it achieved impressive results for the image-to-image translation problem.

Image Discriminator. We implement the discriminator $D_{\mathbf{I}}(\mathbf{I})$ as a PatchGan [52] network mapping from the input image $\mathbf{I}$ to a matrix $\mathbf{Y}_{\mathbf{I}} \in \mathbb{R}^{26 \times 26}$, where $\mathbf{Y}_{\mathbf{I}}[i, j]$ represents the probability of the overlapping patch ij to be real. This discriminator contains less parameters than other conventional discriminators typically used for GANs and enforces high frequency correctness to reduce the blurriness of the generated images.

Pose Detector. Given an image $\mathbf{I}$ of a person, $\Phi(\mathbf{I})$ is a 2D detection network responsible for estimating the skeleton joint locations $\mathbf{p} \in \mathbb{R}^{N \times H \times W}$ in the image plane. $\Phi(\mathbf{I})$ is implemented with the ResNet [193] based network by Zhu *et al.* [243].

4.3.3 Learning the Model

The loss function we define contains three terms, namely an image adversarial loss that pushes the distribution of the generated images to the distribution of the training images, the *conditional pose loss* that enforces the pose of the generated images to be similar to the desired ones, and the *identity loss* that favors to preserve the person identity. We next describe each of these terms.

Image Adversarial Loss. In order to optimize the generator G parameters and learn the distribution of the training data, we perform a standard *min-max strategy game* between the generator and the image discriminator $D_{\mathbf{I}}$. The generator and discriminator are jointly trained with the objective function $\mathcal{L}_{\mathbf{I}}(G, D_{\mathbf{I}}, \mathbf{I}, \mathbf{p})$ where $D_{\mathbf{I}}$ tries to maximize the probability of correctly classifying real and rendered images while G tries to foul the discriminator. Formally, this loss is defined as:

$$\mathcal{L}_{\mathbf{I}}(G, D_{\mathbf{I}}, \mathbf{I}, \mathbf{p}) = \mathbb{E}_{\mathbf{I} \sim p_{\text{data}}(\mathbf{I})}[\log D_{\mathbf{I}}(\mathbf{I})] + \mathbb{E}_{\mathbf{I} \sim p_{\text{data}}(\mathbf{I})}[\log(1 - D_{\mathbf{I}}(G(\mathbf{I}|\mathbf{p})))] \tag{4.2}$$

Conditional Pose Loss. While reducing the *image adversial loss*, the generator must also reduce the error produced by the 2D pose regressor Φ. In this way, the generator not only learns to produce realistic samples but also learns how to generate samples consistent with the desired pose $\mathbf{p}$. This loss is defined by:

$$\mathcal{L}_{\mathrm{P}}(G, \Phi, \mathbf{I}, \mathbf{p}) = \|\Phi(G(\mathbf{I}|\mathbf{p})) - \mathbf{p}\|_2^2 \tag{4.3}$$

Identity Loss. With the two previously defined losses $\mathcal{L}_{\mathbf{I}}$ and $\mathcal{L}_{\mathrm{P}}$ the generator is enforced to generate realistic images of people in a desired position. However, without ground-truth supervision there is no constraint to guarantee that the person identity – *e.g.*, body shape, hair style – in the original and rendered images is the same. In order to preserve person identity, we draw inspiration on the *content-style loss* that was previously introduced in [267] to maintain high perceptual quality in the problem of image style transfer. This loss consists of two main components, one to retain semantic similarity (*content*) and the other to retain texture similarity (*style*). Based on this idea we define two sub-losses that aim at retaining the identity between the input image $\mathbf{I}_{p_o}$ and the rendered image $\mathbf{I}_{p_f}$.

For the *content* term, we argue that the generator should be able to render-back the original image $\mathbf{I}_{p_o}$ given the generated image $\mathbf{I}_{p_f}$ and the original pose $\mathbf{p}_o$, that is, $\hat{\mathbf{I}}_o \approx \mathbf{I}_{p_o}$, where $\hat{\mathbf{I}}_o = G(G(\mathbf{I}_{p_o}|\mathbf{p}_f)|\mathbf{p}_o)$. Nevertheless, even when using PatchGan based discriminators, directly comparing $\mathbf{I}_{p_o}$ and $\hat{\mathbf{I}}_{p_o}$ at a pixel level would struggle to handle high-frequency details leading to overly-smoothed images. Instead, we compare them based on their semantic content. Formally, we define the content loss to be:

$$\mathcal{L}_{\text{Content}} = \|\Psi_z(\mathbf{I}_{p_o}) - \Psi_z(\hat{\mathbf{I}}_{p_o})\|_2^2 \tag{4.4}$$

where $\Psi_z(\cdot)$ represents the activations at the z-th layer of a pretrained network.

In order to retain the *style* of the original image into the rendered ones we enforce the texture around the visible joints of $\mathbf{I}_{p_o}$ and $\mathbf{I}_{p_f}$ to be similar. This involves a first step of extracting, in a differential manner, patches of features around the joints of $\mathbf{I}_{p_o}$ and $\mathbf{I}_{p_f}$. More specifically, let $\Psi_z(\mathbf{I}_{p_o}) \in \mathbb{R}^{C\times H'\times W'}$ be the semantic features of $\mathbf{I}_{p_o}$, and $\mathbf{B}_{p_o} \in \mathbb{R}^{N\times H'\times W'}$ the down-sampled (using average pooling) probability maps associated to the pose $\mathbf{p}_o$. The pose-conditioned patches are computed as:

$$\mathbf{X}_{p_o,i} = \mathbf{B}_{p_o,i} \cdot \Psi_z(\mathbf{I}_{p_o}) \quad \forall i \in \{1, \ldots, N\} \tag{4.5}$$

The representation of a patch style is then captured by the correlation between the different

channels of its hidden representations $\mathbf{X}_{p_o,i}$ using the spatial extend of the feature maps as the expectation. As previously done in [267] this can be implemented by computing the Gram matrix $\mathcal{G}_{p_o,i} \in \mathbb{R}^{C\times C}$ for each patch i, defined as the inner product between the vectorized feature maps of $\mathbf{X}_{p_o,i}$. The *Patch-Style* loss is then computed as the mean square error between visible pairs of Gram matrices of the same joint in both images $\mathbf{I}_{p_o}$ and $\mathbf{I}_{p_f}$:

$$\mathcal{L}_{\text{Patch-Style}} = \frac{1}{N}\sum_{i}^{N}\left(\frac{\mathcal{G}_{p_o,i} - \mathcal{G}_{p_f,i}}{H'W'}\right)^2 \tag{4.6}$$

Finally, we define the identity loss as the weighted sum of the content and style losses:

$$\mathcal{L}_{\text{Id}} = \mathcal{L}_{\text{Content}}(\Psi, \mathbf{I}_{p_o}, \hat{I}_{p_o}) + \lambda\mathcal{L}_{\text{Patch-Style}}(\Psi, \mathbf{I}_{p_o}, \mathbf{I}_{p_f}, \mathbf{p}_o, \mathbf{p}_f), \tag{4.7}$$

where he parameter λ controls the relative importance or the two components.

Full Loss. We take the full loss as a linear combination of all previous loss terms:

$$\begin{aligned}\mathcal{L} = &\mathcal{L}_{\mathbf{I}}(G, D_{\mathbf{I}}, \mathbf{I}_{p_o}, \mathbf{p}_f) + \lambda_{\text{P}}\mathcal{L}_{\text{P}}(G, \Phi, \mathbf{I}_{p_o}, \mathbf{p}_f)\\ &+ \mathcal{L}_{\mathbf{I}}(G, D_{\mathbf{I}}, \mathbf{I}_{p_f}, \mathbf{p}_o) + \lambda_{\text{P}}\mathcal{L}_{\text{P}}(G, \Phi, \mathbf{I}_{p_f}, \mathbf{p}_o)\\ &+ \lambda_{\text{Id}}\mathcal{L}_{\text{Id}} + \lambda_{\text{P}}\mathcal{L}_{\Phi}(\mathbf{I}, \mathbf{p}_o),\end{aligned} \tag{4.8}$$

where $\mathcal{L}_{\Phi}(\mathbf{I}, \mathbf{p}_o) = \|\Phi(\mathbf{I}_{p_o}) - \mathbf{p}_o\|_2^2$ is used to train the pose regressor Φ. Our ultimate goal is to solve:

$$G^{\star} = \arg\min_{G}\max_{D_{\text{I}},\Phi}\mathcal{L} \tag{4.9}$$

Some could argue that the terms $\mathcal{L}_{\mathbf{I}}$ and $\mathcal{L}_{\text{P}}$ for the recovered image $\hat{\mathbf{I}}_{p_o}$ are not required because the same information is expressed by $\mathcal{L}_{\text{Content}}$. However, we experienced that these two terms improved robustness and convergence properties during training.

Implementation Details

In order to reduce the model oscillation and obtain more photo-realistic results we use the learning trick introduced in [269] and replace the negative log likelihood of the adversarial loss by a least square loss. The image features $\Psi_z(\mathbf{I})$ are obtained from a pretrained VGG16 [270] with $z = 7$. We use Adam solver [38] with learning rate of 0.0002 for the generator, 0.0001 for the discriminators and a batch size 12. We train for 300 epochs with a linear decreasing rate after epoch 100. The weights for the loss terms are set to $\lambda_{\text{P}} = 700$ and $\lambda_{\text{Id}} = 0.3$. As in [271], to improve training stability, we update the discriminators using a buffer with the

previous rendered images rather than those generated in the current iteration. During training, the $\mathbf{p}_f$ poses are randomly sampled from those in the training set.

4.3.4 Experimental Evaluation

We verify the effectiveness of our unsupervised GAN model through quantitative and qualitative evaluations. We next describe the dataset we used for evaluation and the results we obtained.

We have evaluated our approach on the publicly available *In-shop Clothes Retrieval Benchmark* of the DeepFashion dataset [7], that contains a large number of clothing images with diverse person poses. Images of the dataset were initially resized to a fixed size of 256×256. We then applied data augmentation with all three possible flips per each image. After that, 2D pose was computed in all images using the Convolutional Pose Machine [194], and images for which it failed were removed from the dataset. From the remaining images, we randomly selected 24,145 for training and 5,000 for test. Test samples are also associated to a desired pose and its corresponding ground truth image, that will be used for quantitative evaluation purposes. Training images are only associated to a desired 2D pose. No ground truth warped image is considered during training.

Quantitative results

Since test samples are annotated with ground truth images under the desired pose, we can quantitatively evaluate the quality of the synthesis. Specifically, we use the metrics considered by previous approaches on multi-view person generation [14,234], namely the Structural Similarity (SSIM) [272] and the Inception Score (IS) [16]. These are fairly standard metrics that focus more on the overall quality of the generated image rather than on the pixel-level similarity between the generated image and the ground truth. Concretely, SSIM models the changes in the structural information and IS give high scores for images with a large semantic content.

In Table 4.1 we report these scores for our approach and two fully supervised methods [234] and [14], when evaluated on the DeepFashion [7] dataset. Two additional implementations of a VAE [273] and a Conditional GAN model [47], reported in [14], are included. It is worth to point that while all methods are evaluated on the same dataset, the test splits in each case are not the same. Therefore, the results on this table should be considered only as indicative. In any event, note that the two metrics indicate that the quality of the synthesis obtained by our unsupervised approach are very similar to the most recent supervised approaches and even outperform previous VAE and conditional GAN implementations.

Method	SSIM ↑	IS ↑
Our Approach	0.747	2.97
Ma *et al.*NIPS'2017 [234]	0.762	3.09
Zhao *et al.*ArXiv'2017 [14]	0.620	3.03
Sohn *et al.*NIPS'2015 [273]*	0.580	2.35
Mirza *et al.*ArXiv'2014 [47]*	0.590	2.45

Table 4.1: **Quantitative evaluation on the DeepFashion dataset.** SSIM and IS for our unsupervised approach and four supervised state-of-the-art methods. For all measures, the higher is better. '*' indicates that these results were taken from [14]. Note: These results are just indicative, as the test splits in previous approaches are not available and may differ between the different methods of the table. Nevertheless, note that the quantitative results put our unsupervised approach on a par with other supervised approaches.

Qualitative results

We next present and discuss a series of qualitative results that will highlight the main characteristics of the proposed approach, including its ability to generalize to novel poses, to hallucinate image patches not observed in the original image and to render textures with high-frequency details.

In Fig. 4.1 we observe all these characteristics. First, note the ability of our GAN model to generalize to desired poses very different from that in the original image. In this case given a frontal image of the upper body of a woman, we show some of the generated images in which her pose is rotated by 180 deg. In the right-most image of this example, the network is also able to hallucinate the two legs, not seen in the original image (despite not rendering the skirt). For this particular example, the network convincingly renders the high frequency details of the blouse. This is a very important characteristic of our model, and is a direct consequence of the loss function we have designed, and in particular of the term $\mathcal{L}_{\text{Patch-Style}}$ in Eq. (4.6) that aims at retaining the texture details of the original image into the generated one. This is in contrast to most of the renders generated by other GAN models [14, 233, 234], which typically wash out texture details.

Fig. 4.3 presents another series of results obtained with our model. In this case, each synthetically generated image is accompanied by the ground truth. Note again, the number of complex examples that are successfully addressed. Several cases show the hallucination of frontal poses from original poses facing back (or vice versa). Also are worth to mention those examples where the original image is in a side position with only one arm being observed, and the desired pose is either frontal of backwards, having to hallucinate both arms. Some of the textures of the t-shirts have very high frequency patterns and textures (example 4-th row/2-nd column, examples 6-th row) that are convincingly rendered under new poses.

Figure 4.3: **Test results on the DeepFashion [7] dataset.** Each test sample is represented by 4 images: input image, 2D desired pose, synthesized image and ground truth.

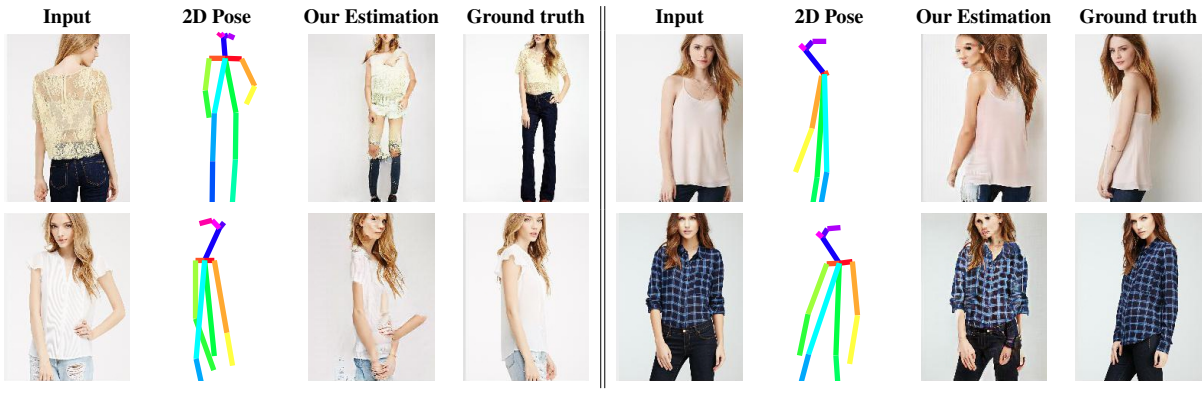


Figure 4.4: **Test failures on the DeepFashion [7] dataset.** We represent four different types of errors that typically occur in the failure cases (see text for details).

Figure 4.5: **L1 vs identity loss.** Synthetic samples obtained by our model when it is trained with L1 loss and conditioned with the same inputs as in Fig. 4.1. The first five columns correspond to $\hat{\mathbf{I}}_{p_f}$, and the last column is the cycle image $\hat{\mathbf{I}}_{p_o}$. Comparing these results with those of Fig. 4.1 it becomes clear that the L1 loss is not able to capture the person identity.

Failure Cases. Tackling such an unconstrained problem in a fully unsupervised manner causes a number of errors. We have roughly split them into four categories which we summarize in Fig. 4.4. The first type of error (top-left) is produced when textures in the original image are not correctly mapped onto the generated image. In this case, the partially observed dark trousers are transferred to a lower leg, resembling boots. In the top-right example, the face of the original image is not fully washed out in the new generated image. In the bottom-left we show a type of error which we denote as *geometric error*, where the pose of the original image is not properly transferred to the target image. The bottom-right image shows an example in which a part of the body in the original image (hand) is mapped as a texture in the synthesized one.

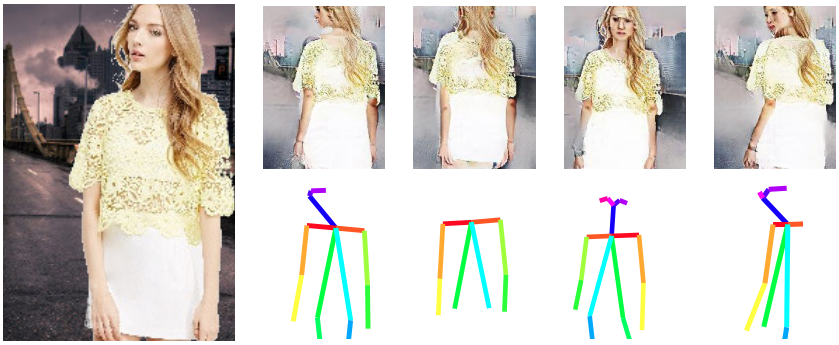


Figure 4.6: **Testing on images with background.** Given the original image of a person with background on the left and a desired body pose defined by a 2D skeleton (bottom-row), the model generates the person under that pose shown in the top-row. Albeit our model is trained with images with no background it does generalize fairly well to this situation (compare with the results of Fig. 4.1).

Ablation Study. Each component is crucial for the proper performance of the system. $D_{\mathbf{I}}$ and $L_{\mathbf{I}}$ constrain the system to generate realistic images; Φ and L_{P} ensure the generator conditions the image generation to the given pose; and Ψ and L_{Id} force the generator to preserve the input image texture. Removing any of these elements would damage our network. For instance, Fig. 4.5 shows the results when replacing L_{Id} by the standard L1 loss used by most state-of-the-art GAN works. As it can observed in the last column of the figure, although $\hat{\mathbf{I}}_{p_o}$ is preserving the low frequency texture of the original image, the person identity in $\mathbf{I}_{p_f}$ is lost and all results tend to converge to a mean brunette woman with white t-shirt and blue jeans.

Images with Background. To further test the limits of our model, Fig. 4.6 presents an evaluation of the model performance when the input image contains background. Surprisingly, although the model has no loss on background consistency nor was trained with images with background, the results are still very consistent. The person is quite correctly rendered, while the background is over-smoothed. To become robust to background would require more complex datasets and specialized loss functions.

4.4 Unsupervised Face Image Synthesis and Animation

Once studied the change of view and pose of a clothed person we now focus on the face. And more specifically to the task of automatically and smoothly change the facial expression from a single image. As GANs have become more prevalent, this task has experienced significant advances, with architectures such as StarGAN [11], which is able not only to synthesize novel

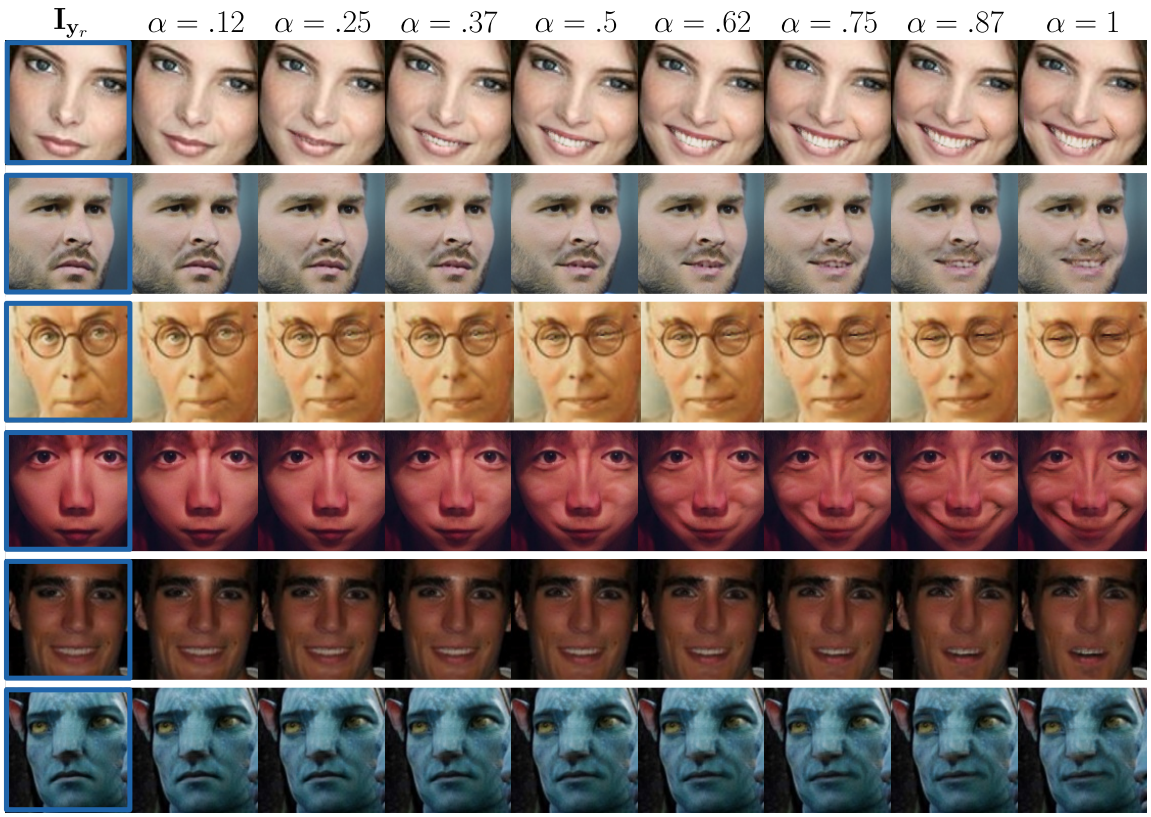


Figure 4.7: **Facial animation from a single image**. We propose GANimation, an anatomically coherent approach that is not constrained to a discrete number of expressions and can animate the face in a given image and render novel expressions in a continuum. In these examples, we are given solely the left-most input image $\mathbf{I}_{\mathbf{y}_r}$ (highlighted by a blue square), and the parameter α shown on the top denotes the degree of activation of the target Action Units involved in a smiling-like expression. Additionally, our system can handle images with complex illumination and non-human skin textures, such as the example in the bottom row.

expressions, but also to change other attributes of the face, such as age, hair color or gender. Despite its generality, StarGAN can only change a particular aspect of a face among a discrete number of attributes defined by the annotation granularity of the dataset. For instance, for the facial expression synthesis task, [11] is trained on the RaFD [274] dataset which has only 8 binary labels for facial expressions, namely sad, neutral, angry, contemptuous, disgusted, surprised, fearful and happy, respectively. The generation possibilities of [11] are, in this case, limited by these eight expression categories.

Facial expressions, however, are the result of the combined and coordinated action of facial muscles that cannot be categorized in a discrete and low number of classes. Ekman and Friesen [275] developed the Facial Action Coding System (FACS) for describing facial expressions in terms of the so-called Action Units (AUs), which are anatomically related to the contrac-

tions of specific facial muscles. Although the number of Action Units is relatively small (30 AUs were found to be anatomically related to the contraction of specific facial muscles), more than 7,000 different AU combinations have been observed [276]. For example, the facial expression for *fear* is generally produced with the following activation state: Inner Brow Raiser (AU1), Outer Brow Raiser (AU2), Brow Lowerer (AU4), Upper Lid Raiser (AU5), Lid Tightener (AU7), Lip Stretcher (AU20) and Jaw Drop (AU26) [277]. Depending on the magnitude of each AU, the expression will transmit the emotion of fear to a greater or lesser extent.

In this section we aim at building a model for synthetic facial animation with the level of expressiveness of FACS, and being able to generate anatomically-aware expressions in a continuous domain, without the need to pre-compute the position of facial landmarks in the input images [278]. For this purpose we leverage on the recent EmotioNet dataset [279], which consists of one million images (we use 200,000 of them) of facial expressions of emotion in the wild annotated with discrete AUs' activation[1]. We build a GAN architecture which, instead of being conditioned with images of a specific domain as in [11], it is conditioned on a one-dimensional vector indicating the presence/absence and the magnitude of each Action Unit. We train this architecture in an weakly supervised manner that only requires images with their activated AUs. To circumvent the need for pairs of training images of the same person under different expressions, we split the problem in two main stages. First, we consider an AU-conditioned bidirectional adversarial architecture which, given a single training photo, initially renders a new image under the desired expression. This synthesized image is then rendered-back to the original expression, hence being directly comparable to the input image. We incorporate very recent losses to enforce the photo-realism of the generated image. Additionally, our system also goes beyond state of the art in that it can handle images under changing backgrounds and illumination conditions. We achieve this by means of a self-learned attention layer that focuses the action of the network only in those regions of the image that are relevant to convey the novel expression.

As a result, we build an anatomically coherent facial expression synthesis method, able to render images in a continuous domain, and which can handle images in the wild with complex backgrounds and illumination conditions. As we will show in the results subsection, it compares favorably to other conditioned-GANs schemes, both in terms of the visual quality of the results, and the possibilities of generation. Fig. 4.7 shows some example of the results we obtain, in which given one input image, we gradually change the magnitude of activation of the AUs used to produce a smile. We have also particularly analyzed the role of the attention mechanism we propose, which is a key ingredient of our architecture, and brings robustness to several artifacts. We show that besides yielding robustness to cluttered backgrounds it is also effective to handle

[1]The dataset was re-annotated with [280] to obtain continuous activation annotations.

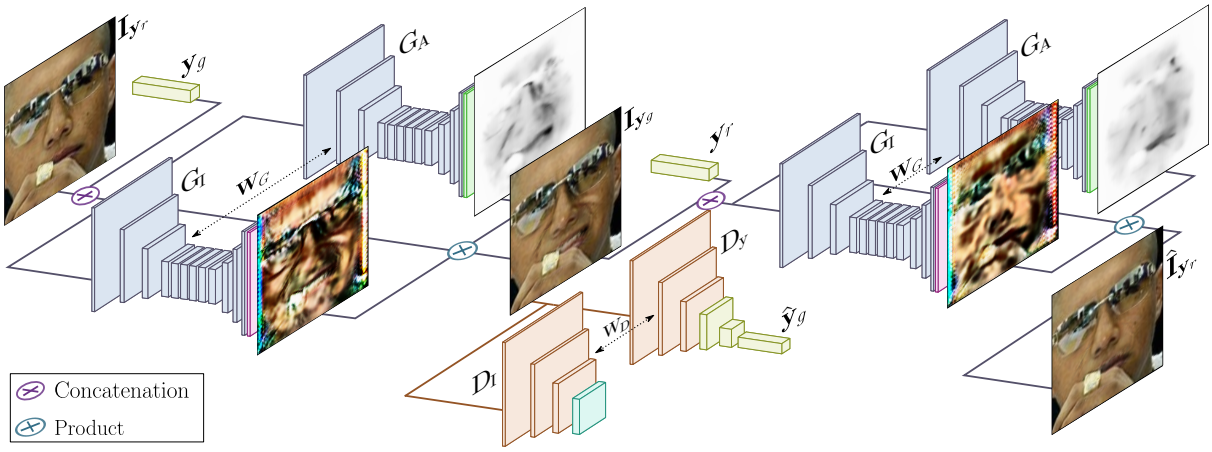


Figure 4.8: **GANimation overview.** The architecture of GANimation consists of three main blocks: a generator G to regress attention and color masks; a critic D_{I} to evaluate the quality of the generated image and its photo-realism; and finally, an expression estimator D_{y} to penalize differences between the desired conditioning expression $\mathbf{y}_g$ and its fulfillment $\hat{\mathbf{y}}_g$. It is worth noting that our scheme does not require supervision, *i.e.*, neither pairs of images of the same person under different expressions, nor the target image $\mathbf{I}_{\mathbf{y}_g}$ are assumed to be known.

partial occlusions of the face. Finally, we also provide a user study to assess the quality of the generated results.

In summary, this section main contributions are: 1) a novel generative model for face animation in the wild that can be trained in a weakly supervised manner. It advances current works which, so far, had only addressed the problem for discrete emotions category editing and portrait images; and 2) a plug-and-play attention mechanisms for image editing networks making them robust to changing backgrounds and lighting conditions for images in the wild.

4.4.1 Problem Formulation

Let us define an input RGB image as $\mathbf{I}_{\mathbf{y}_r} \in \mathbb{R}^{H \times W \times 3}$, which represents the cropped face of a subject under an arbitrary expression. Every gesture expression is encoded by means of a set of N Action Units $\mathbf{y}_r = (y_1, \ldots, y_N)^\top$, where each y_n denotes a normalized value between 0 and 1 to module the magnitude of the n-th Action Unit. This type of continuous representation is a key ingredient of our design, as a natural interpolation can be done between different expressions, allowing to render a wide range of realistic and smoothly changing facial expressions.

Our aim is to learn a mapping $\mathcal{M}$ to translate $\mathbf{I}_{\mathbf{y}_r}$ into an output image $\mathbf{I}_{\mathbf{y}_g}$ conditioned on an action-unit target $\mathbf{y}_g$, *i.e.*, we seek to estimate the mapping $\mathcal{M} : (\mathbf{I}_{\mathbf{y}_r}, \mathbf{y}_g) \rightarrow \mathbf{I}_{\mathbf{y}_g}$. To

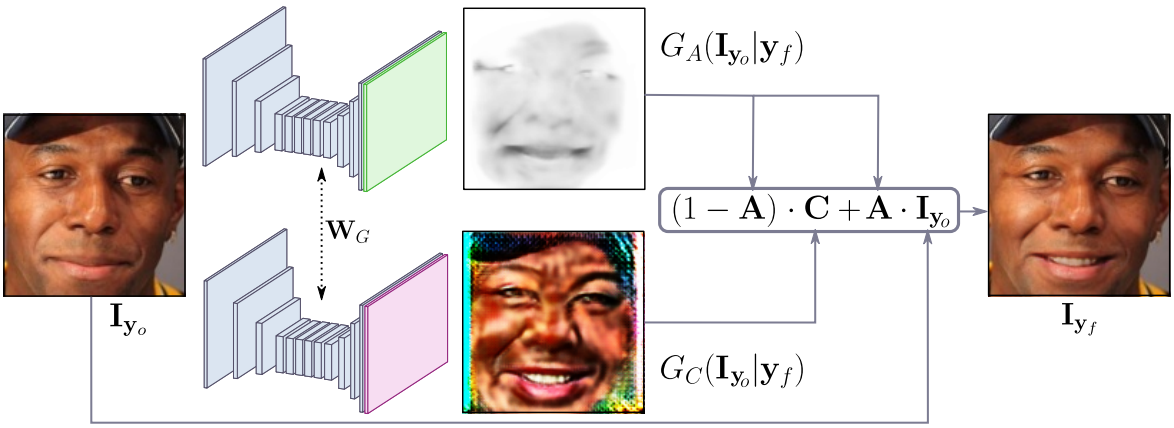


Figure 4.9: **Attention-based generator.** Given an input image and the target expression, the generator regresses an attention mask $\mathbf{A}$ and an RGB color transformation $\mathbf{C}$ over the entire image. The attention mask defines a per pixel intensity specifying to which extend each pixel of the original image will contribute in the final rendered image.

this end, we propose to train $\mathcal{M}$ in a weakly supervised manner, using M training triplets $\{\mathbf{I}_{\mathbf{y}_r}^{(m)}, \mathbf{y}_r^{(m)}, \mathbf{y}_g^{(m)}\}_{m=1}^{M}$, where the target vectors $\mathbf{y}_g^m$ are randomly generated. Importantly, we neither require pairs of images of the same subject under different expressions, nor the expected target image $\mathbf{I}_{\mathbf{y}_g}$ to be known.

4.4.2 Anatomically-aware Generative Model

This subsection describes our novel approach to generate photo-realistic conditioned images, which, as shown in Fig. 4.8, consists of two main modules. On the one hand, a generator $G(\mathbf{I}_{\mathbf{y}_r}|\mathbf{y}_g)$ is trained to realistically transform the facial expression in image $\mathbf{I}_{\mathbf{y}_r}$ to the desired $\mathbf{y}_g$. Note that G is applied twice, first to map the input image $\mathbf{I}_{\mathbf{y}_r} \rightarrow \mathbf{I}_{\mathbf{y}_g}$, and then to render it back $\mathbf{I}_{\mathbf{y}_g} \rightarrow \hat{\mathbf{I}}_{\mathbf{y}_r}$. On the other hand, we use a WGAN-GP [44] based critic $D_{\mathrm{I}}(\mathbf{I}_{\mathbf{y}_g})$ to evaluate the quality of the generated image and an expression estimator $D_{\mathrm{y}}(\mathbf{I}_{\mathbf{y}_g})$ to penalize differences between the desired and generated expression. We next describe in detail each one of these blocks.

Generator. Let G be the generator block. Since it will be applied bidirectionally (*i.e.*, to map input image to desired expression and vice-versa) in the following discussion we use subscripts o and f to indicate *origin* and *final*.

Given the image $\mathbf{I}_{\mathbf{y}_o} \in \mathbb{R}^{H\times W\times 3}$ and the N-vector $\mathbf{y}_f$ encoding the desired expression, we form the input of the generator as a concatenation $(\mathbf{I}_{\mathbf{y}_o}, \mathbf{y}_o) \in \mathbb{R}^{H\times W\times (N+3)}$, where $\mathbf{y}_o$ is represented as N arrays of size $H \times W$.

One essential component of our system is to make G focus only on those regions of the

image that are responsible of synthesizing the novel expression and keep the rest elements of the image such as hair, glasses, hats or jewellery untouched. For this purpose, we have embedded an attention mechanism into the generator. Concretely, instead of regressing a full image, our generator outputs two masks, a color mask $\mathbf{C}$ and an attention mask $\mathbf{A}$. The final image can be obtained as:

$$\mathbf{I}_{\mathbf{y}_f} = (1 - \mathbf{A}) \cdot \mathbf{C} + \mathbf{A} \cdot \mathbf{I}_{\mathbf{y}_o} \ , \tag{4.10}$$

where $\mathbf{A} = G_A(\mathbf{I}_{\mathbf{y}_o}|\mathbf{y}_f) \in [0,1]^{H \times W}$ and $\mathbf{C} = G_C(\mathbf{I}_{y_o}|\mathbf{y}_f) \in \mathbb{R}^{H \times W \times 3}$. The mask $\mathbf{A}$ indicates to which extend each pixel of $\mathbf{C}$ contributes to the output image $\mathbf{I}_{\mathbf{y}_f}$. In this way, the generator does not need to render static elements, and can focus exclusively on the pixels defining the facial movements, leading to sharper and more realistic synthetic images. This process is depicted in Fig. 4.9.

Conditional Critic. This block is a network trained to evaluate the generated images in terms of their photo-realism. The structure of $D_{\mathrm{I}}(\mathbf{I})$ resembles that of the PatchGan [52] network mapping from the input image $\mathbf{I}$ to a matrix $\mathbf{Y}_{\mathrm{I}} \in \mathbb{R}^{H/2^6 \times W/2^6}$, where $\mathbf{Y}_{\mathrm{I}}[i,j]$ is used as a partial function to compute the EMD between the distributions of real image patches and the overlapping patch ij of the generated image.

Expression Estimator. Given an image $\mathbf{I}$ of a face, $D_{\mathrm{y}}(\mathbf{I})$ is an expression regression network responsible for estimating the AUs' activation $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_N)^\top$ in the image. Similar to the conditional critic, its structure resembles that of PatchGan. To reduce the number of parameters of the model, $D_{\mathrm{y}}(\mathbf{I})$ is implemented on top of the conditional critic as an auxiliary head sharing the weights of the first five layers.

4.4.3 Learning the Model

The parameters of the generator, conditional critic and expression estimator are simultaneously estimated. For this purpose we define a loss function made of four terms, namely an *image adversarial loss* [43] with the modification proposed by Gulrajani *et al.* [44] that pushes the distribution of the generated images to the distribution of the training images; the *attention loss* to drive the attention masks to be smooth and prevent them from saturating; the *conditional expression loss* that conditions the expression of the generated images to be similar to the desired one; and the *identity loss* that favors to preserve the person texture identity. In the following we describe these losses:

Image Adversarial Loss. In order to learn the parameters of the generator G, we use the modification of the standard GAN algorithm [40] proposed by WGAN-GP [44] as in Sec. 3.6. As a quick reminder for the reader, the original GAN formulation is based on the JS divergence loss

function and aims to maximize the probability of correctly classifying real and rendered images while the generator tries to foul the discriminator. This loss is potentially not continuous with respect to the generator's parameters and can locally saturate leading to vanishing gradients in the discriminator. This is addressed in WGAN [43] by replacing JS with the continuous EMD. To maintain a Lipschitz constraint, WGAN-GP [44] proposes to add a gradient penalty for the critic network computed as the norm of the gradients with respect to the critic input.

Formally, let $\mathbf{I}_{\mathbf{y}_o}$ be the input image with the initial condition $\mathbf{y}_o$, $\mathbf{y}_f$ the desired final condition, $\mathbb{P}_o$ the data distribution of the input image, and $\mathbb{P}_{\widetilde{\mathbf{I}}}$ the random interpolation distribution. Then, the *critic loss* we use is:

$$\begin{aligned}\mathcal{L}_{\mathrm{I}}(G, D_{\mathrm{I}}, \mathbf{I}_{\mathbf{y}_o}, \mathbf{y}_f) = &- \mathbb{E}_{\mathbf{I}_{\mathbf{y}_o}\sim\mathbb{P}_o}[D_{\mathrm{I}}(G(\mathbf{I}_{\mathbf{y}_o}|\mathbf{y}_f))] + \mathbb{E}_{\mathbf{I}_{\mathbf{y}_o}\sim\mathbb{P}_o}\left[D_{\mathrm{I}}(\mathbf{I}_{\mathbf{y}_o})\right] \\ &- \lambda_{\mathrm{gp}}\mathbb{E}_{\widetilde{I}\sim\mathbb{P}_{\widetilde{\mathbf{I}}}}\left[(\|\nabla_{\widetilde{I}} D_{\mathbf{I}}(\widetilde{\mathbf{I}})\|_2 - 1)^2\right],\end{aligned} \tag{4.11}$$

where λ_{gp} is a penalty coefficient.

Attention Loss. When training the model we do not have ground-truth annotation for the attention masks $\mathbf{A}$. Similarly as for the color masks $\mathbf{C}$, they are learned from the resulting gradients of the critic module and the rest of the losses. However, the attention masks can easily saturate to 1 which makes that $\mathbf{I}_{\mathbf{y}_o} = G(\mathbf{I}_{\mathbf{y}_o}|\mathbf{y}_f)$, that is, the generator has no effect. To prevent this situation, we regularize the mask with a weight penalty. Additionally, to enforce a smooth spatial color transformation when combining the regions of the input image and those of the color transformation $\mathbf{C}$, we perform a *Total Variation Regularization* over $\mathbf{A}$. The attention loss can therefore be defined as:

$$\mathcal{L}_{\mathbf{A}}(G, \mathbf{I}_{\mathbf{y}_o}, \mathbf{y}_f) = \lambda_{\mathrm{TV}}\mathcal{L}_{\mathrm{TV}}(\mathbf{A}) + \mathbb{E}_{\mathbf{I}_{\mathbf{y}_o}\sim\mathbb{P}_o}\left[\|\mathbf{A}\|\right], \tag{4.12}$$

where $\mathbf{A} = G_A(\mathbf{I}_{\mathbf{y}_o}|\mathbf{y}_f)$ and $\mathbf{A}_{i,j}$ represents the $[i, j]$ entry of $\mathbf{A}$. λ_{TV} is a penalty coefficient for the mask smoothing, being the corresponding loss defined as:

$$\mathcal{L}_{\mathrm{TV}}(\mathbf{A}) = \sum_{i,j}^{H,W}\left[(\mathbf{A}_{i+1,j} - \mathbf{A}_{i,j})^2 + (\mathbf{A}_{i,j+1} - \mathbf{A}_{i,j})^2\right].$$

Conditional Expression Loss. While reducing the *image adversarial loss*, the generator must also reduce the error produced by the AUs' regression head on top of D. In this way, G not only learns to render realistic samples but also learns to satisfy the target facial expression encoded by $\mathbf{y}_f$. This loss is defined with two components: an AUs regression loss with fake images used to optimize G, and an AUs regression loss of real images used to learn the regression head on

top of D. This loss is computed as:

$$\mathcal{L}_{\text{y}}(G, D_{\text{y}}, \mathbf{I}_{\mathbf{y}_o}, \mathbf{y}_o, \mathbf{y}_f) = \mathbb{E}_{\mathbf{I}_{\mathbf{y}_o} \sim \mathbb{P}_o} \left[\| D_{\text{y}}(\mathbf{I}_{\mathbf{y}_o}) - \mathbf{y}_o \|_2^2 \right] + \mathbb{E}_{\mathbf{I}_{\mathbf{y}_o} \sim \mathbb{P}_o} \left[\| D_{\text{y}}(G(\mathbf{I}_{\mathbf{y}_o} | \mathbf{y}_f))] - \mathbf{y}_f \|_2^2 \right] . \tag{4.13}$$

Identity Loss. With the previously defined losses the generator is enforced to generate photo-realistic face transformations. However, without ground-truth supervision, there is no constraint to guarantee that the face in both the input and output images correspond to the same person. Using a *cycle consistency loss* [243] we force the generator to maintain the identity of each individual by penalizing the difference between the original image $\mathbf{I}_{\mathbf{y}_o}$ and its reconstruction. The *identity loss* $\mathcal{L}_{\text{idt}}(G, \mathbf{I}_{\mathbf{y}_o}, \mathbf{y}_o, \mathbf{y}_f)$ is defined as:

$$\mathbb{E}_{\mathbf{I}_{\mathbf{y}_o} \sim \mathbb{P}_o} \left[\| G(G(\mathbf{I}_{\mathbf{y}_o} | \mathbf{y}_f) | \mathbf{y}_o) - \mathbf{I}_{\mathbf{y}_o} \|_1 \right] . \tag{4.14}$$

To produce realistic images it is critical for the generator to model both low and high frequencies. Our *PatchGan* based critic D_{I} already enforces high-frequency correctness by restricting our attention to the structure in local image patches. To also capture low-frequencies it is sufficient to use l_1-norm. In preliminary experiments, we also tried replacing l_1-norm with a more sophisticated *Perceptual* loss [268], although we did not observe improved performance.

Full Loss. To generate the target image $\mathbf{I}_{\mathbf{y}_g}$, we build a loss function $\mathcal{L}$ by linearly combining all previous partial losses:

$$\begin{aligned} \mathcal{L} = & \mathcal{L}_{\text{I}}(G, D_{\text{I}}, \mathbf{I}_{\mathbf{y}_r}, \mathbf{y}_g) + \lambda_{\text{y}} \mathcal{L}_{\text{y}}(G, D_{\text{y}}, \mathbf{I}_{\mathbf{y}_r}, \mathbf{y}_r, \mathbf{y}_g) \\ & + \lambda_{\text{A}} \left(\mathcal{L}_{\text{A}}(G, \mathbf{I}_{\mathbf{y}_g}, \mathbf{y}_r) + \mathcal{L}_{\text{A}}(G, \mathbf{I}_{\mathbf{y}_r}, \mathbf{y}_g) \right) \\ & + \lambda_{\text{idt}} \mathcal{L}_{\text{idt}}(G, \mathbf{I}_{\mathbf{y}_r}, \mathbf{y}_r, \mathbf{y}_g), \end{aligned} \tag{4.15}$$

where λ_{A}, λ_{y} and λ_{idt} are the hyper-parameters that control the relative importance of every loss term. Finally, we can define the following minimax problem:

$$G^\star = \arg \min_G \max_{D \in \mathcal{D}} \mathcal{L} \ , \tag{4.16}$$

where $G^\star$ draws samples from the data distribution. Additionally, we constrain our discriminator D to lie on $\mathcal{D}$, that represents the set of 1-Lipschitz functions.

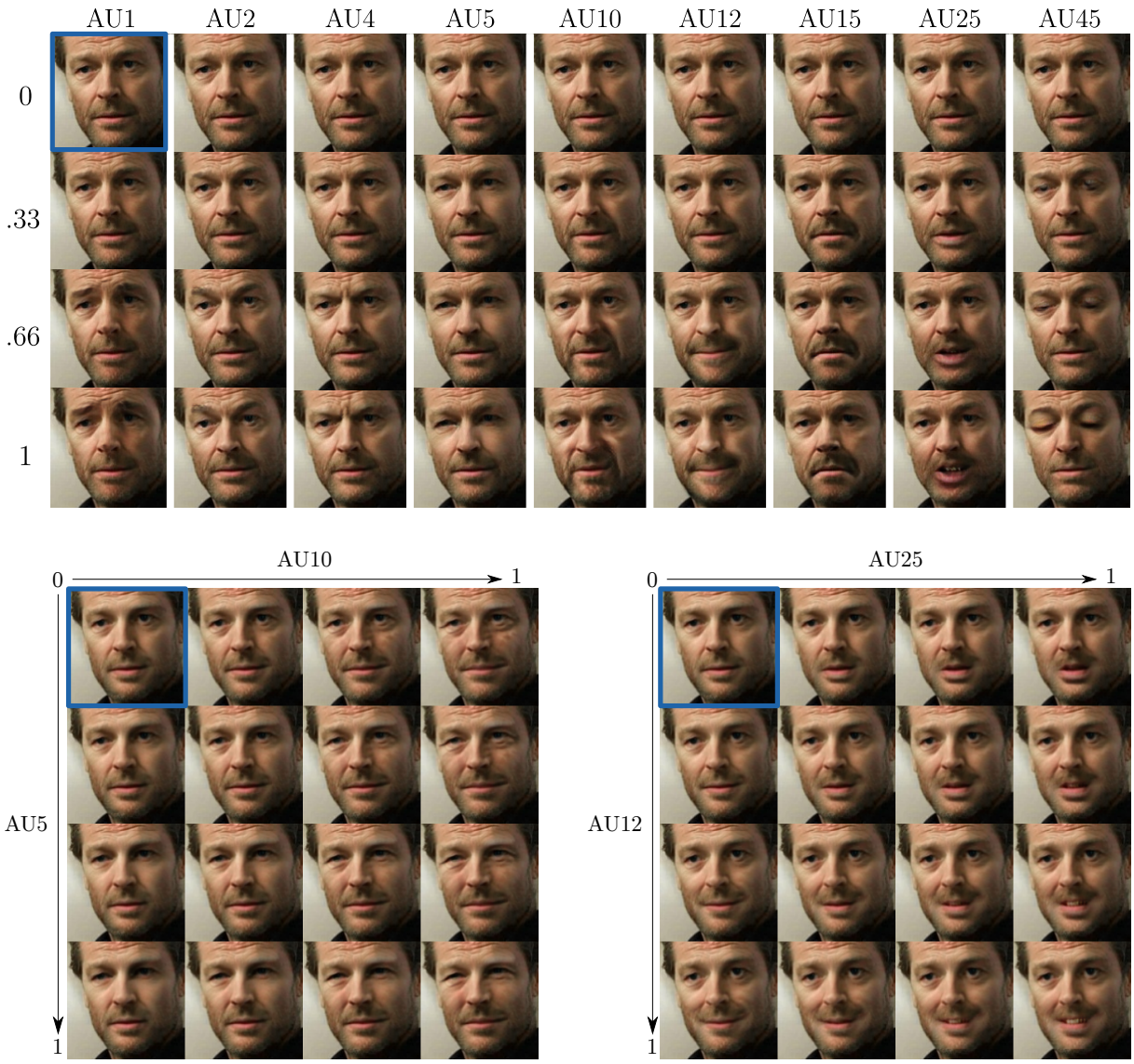


Figure 4.10: **Single and dual-AU edition. Top:** Single AUs are activated at increasing levels of intensity (from .33 to 1). The first row corresponds to a zero intensity application of the AU which correctly produces the original image in all cases. **Bottom:** For every grid two specific AUs are activated at increasing levels of intensity (from 0 to 1). **Left:** Case in which the activation areas of the AUs (#10 and #5) do not overlap. **Right:** Both AUs activate overlapping areas of the face.

Implementation Details

Our generator builds upon the variation of the network from Johnson *et al.* [268] proposed by [243] as it proved to achieve impressive results for image-to-image mapping. We have slightly modified it by substituting the last convolutional layer with two parallel convolutional

Figure 4.11: **Attention model.** Details of the intermediate color mask **C** (first row) and the attention mask **A** (second row). The images in the bottom row are the synthesized expressions. Darker regions of the attention mask **A** show those areas of the image more relevant for each specific AU. Brighter areas are retained from the original image.

layers, one to regress the color mask **C** and the other to define the attention mask **A**. We also observed that changing batch normalization in the generator by instance normalization improved training stability. For the critic we have adopted the *PatchGan* architecture of [52], but removing feature normalization. Otherwise, when computing the gradient penalty, the norm of the critic's gradient would be computed with respect to the entire batch and not with respect to each input independently as is required by WGAN-GP.

The model is trained on the EmotioNet dataset [279]. We use a subset of 200,000 samples (over 1 million) to reduce training time. We use Adam [38] with learning rate of 0.0001, beta1 0.5, beta2 0.999 and batch size 25. We train for 30 epochs and linearly decay the rate to zero over the last 10 epochs. Every five optimization steps of the critic network we perform a single optimization step of the generator. The weight coefficients for the loss terms in Eq. (4.15) are set to $\lambda_{\text{gp}} = 10$, $\lambda_{\text{A}} = 0.1$, $\lambda_{\text{TV}} = 0.0001$, $\lambda_{\text{y}} = 4000$, $\lambda_{\text{idt}} = 10$. To improve stability we tried updating the critic using a buffer with generated images in different updates of the generator, as proposed in [271], but we did not observe performance improvement.

Several design choices (*e.g.*, sharing part of the weights between the conditional critic and the expression estimator) were done in order to fit the model into a single Nvidia ® GTX 1080 Ti GPU with 11GB of RAM. The model is trained in two days on the 200,000 EmotioNet dataset samples. During testing only the regressors are necessary, and hence the size of the model is reduced to 813 MB. Inference can be done at 66 FPS with an Nvidia ® GTX 1080 Ti GPU.

4.4.4 Experimental Evaluation

In this subsection we provide a thorough evaluation of the proposed architecture. Concretely, we

evaluate GANimation's ability for single and multiple AUs editing, for discrete and continuous emotion editing, and compare it with existing techniques. We also provide a detailed analysis of the attention mechanism. Finally, we discuss the model's ability to deal with occlusions and its limitations and failure cases.

It is worth pointing out that in some of the experiments the input faces are not cropped. In these cases we first use an off-the-shelf detector[2] to localize and crop the face, apply the expression transformation to that area with Eq. (4.10), and place the generated face back into its original image position. The attention mechanism is very helpful to process relatively high resolution images and a render smooth transitions between the morphed cropped faces and the original image.

Single Action Units Edition

We first evaluate our model's ability to activate AUs at different intensities while preserving the person's identity. Fig. 4.10-top shows a subset of 9 AUs individually transformed with four levels of intensity (0, .33, .66, 1). For the case of 0 intensity it is desired not to change the corresponding AU. The model properly handles this situation and generates an identical copy of the input image for every case. The ability to apply an identity transformation is essential to ensure that non-desired facial movement is not be introduced.

For the cases with non-zero AU intensity, it can be observed how each AU is progressively accentuated. Note the difference between generated images at intensity 0 and 1. The model convincingly renders complex facial movements which in most cases are difficult to distinguish from real images. It is also worth mentioning that the independence of facial muscle clusters is properly learned by the generator. For instance, AUs relative to the eyes and the upper-half part of the face (AUs 1, 2, 4, 5, 45) do not affect the muscles of the mouth. Equivalently, mouth related transformations (AUs 10, 12, 15, 25) do not affect eyes nor eyebrow muscles.

Fig. 4.11 shows, for the same experiment, the attention $\mathbf{A}$ and color $\mathbf{C}$ masks that produced the final result $\mathbf{I}_{\mathbf{y}_g}$. Note how the model has learned to focus its attention (darker area) onto the corresponding AU in a weakly supervised manner. In this way, it relieves the color mask from having to accurately regress each pixel value. Only the pixels relevant to convey the expression change are carefully estimated, the rest are just set to noise. For example, the attention is clearly obviating background pixels allowing to directly copy them from the original image. This is paramount to later being able to handle images in the wild with complex backgrounds (see Subsec. 4.4.4).

[2]We use the face detector from https://github.com/ageitgey/face_recognition.

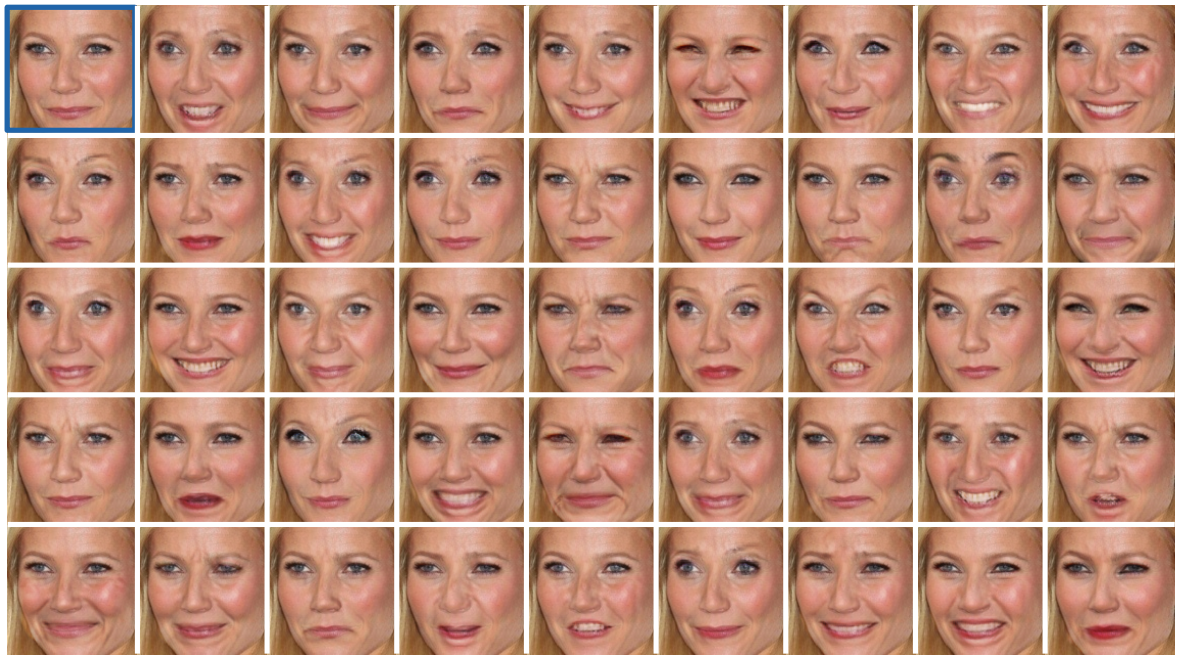


Figure 4.12: **Sampling the face expression distribution space.** As a result of applying the AU-parametrization through the vector $\mathbf{y}_g$, we can synthesize, from the same source image $\mathbf{I}_{\mathbf{y}_r}$, a large variety of photo-realistic images.

Two Action Units Edition

In the following we evaluate the ability of our model to simultaneously activate two actions units. The model must not only be able to activate the desired AUs but also combine them in a realistic manner. The results of this experiment are shown in Fig. 4.10-Bottom. The left grid of the figure shows the case when the two AUs activate different areas of the face (AUs 5 is related to the chicks and 10 to the eyelids). In this case, since the muscles related to each AU are different, their effects are independent from one another. A more difficult case occurs when both AUs share facial muscles, see Fig. 4.10-Bottom-Right. In this specific case, when only activating $AU12$ (left column) the model draws a smile, but when we also activate $AU25$, in charge of controlling the distance between lips, the model produces a smile with the mouth open. Note that the generator hallucinates the teeth that would be visible when smiling with the lips apart.

Simultaneous Edition of Multiple AUs

We next push the limits of the GANimation model and evaluate it in the task of editing multiple AUs. Additionally, we also assess its ability to interpolate between two expressions. The results of this experiment are shown in Fig. 4.7 of the Introduction subsection. The first column is the

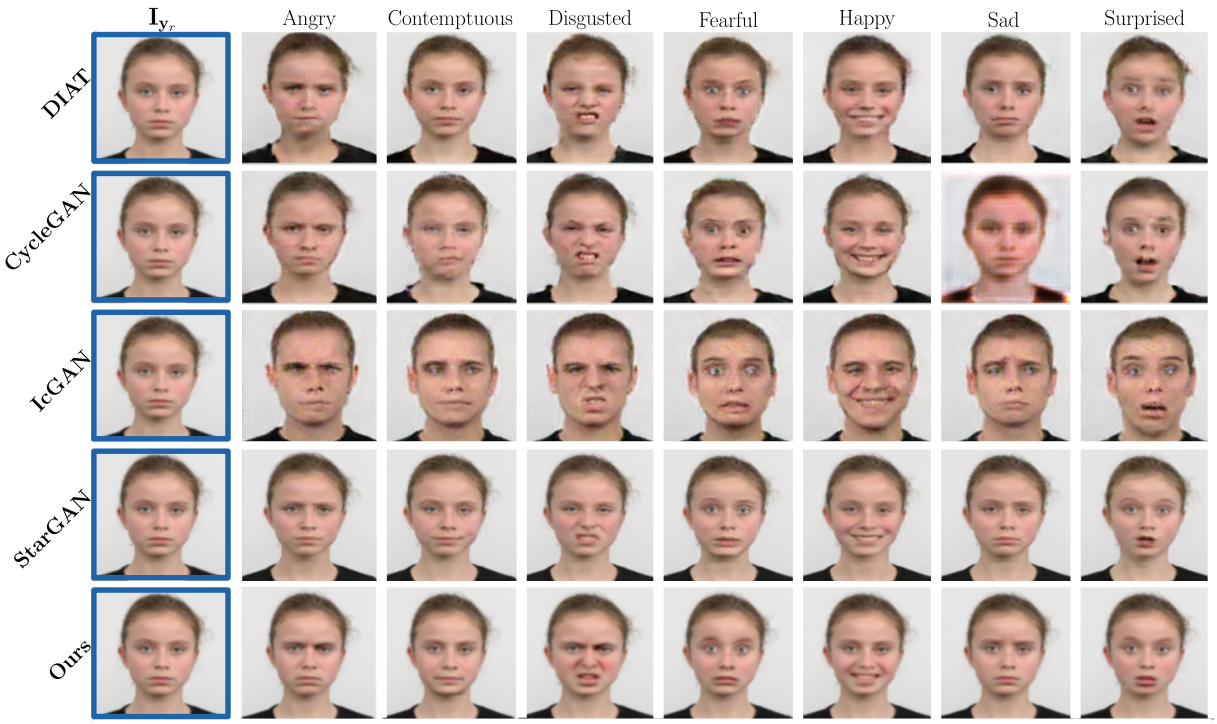


Figure 4.13: **Qualitative comparison with state-of-the-art.** Facial expression synthesis results for: DIAT [8], CycleGAN [9], IcGAN [10] and StarGAN [11]; and our GANimation. In all cases, we represent the input image and seven different facial expressions. As it can be seen, our solution produces the best trade-off between visual accuracy and spatial resolution. Some of the results of StarGAN [11], the best current approach, show certain level of blur. Images of previous models were taken from [11].

original image with expression $\mathbf{y}_r$, and the right-most column is a synthetically generated image conditioned on a target expression $\mathbf{y}_g$. The rest of columns result from evaluating the generator conditioned with a linear interpolation of the original and target expressions: $\alpha\mathbf{y}_g + (1-\alpha)\mathbf{y}_r$. The outcomes show a very remarkable smooth and a consistent transformation across frames. We have intentionally selected challenging samples to show the robustness to complex lighting conditions and even, as in the case of the avatar, to non-real data distributions which were not previously seen by the model. These results are encouraging to further extend the model to video generation [281–285] in future works.

High Expressions Variability

Given a single image, we next use GANimation to produce a wide range of anatomically feasible face expressions while conserving the person's identity. In Fig. 4.12 all faces are the result of conditioning the input image in the top-left corner with a desired face configuration defined

Method	ACD ↓	IS ↑	User Preference ↑
GANimation	0.31	**1.48**	**56%**
StarGaN	**0.29**	1.41	44%

Table 4.2: **Quantitative comparison with StarGAN [11].** The table reports the results of three metrics (described in the text): *Face Distance* (ACD [15], the lower the better), *Inception Score* (IS [16], the higher the better) and user preference (the higher the better).

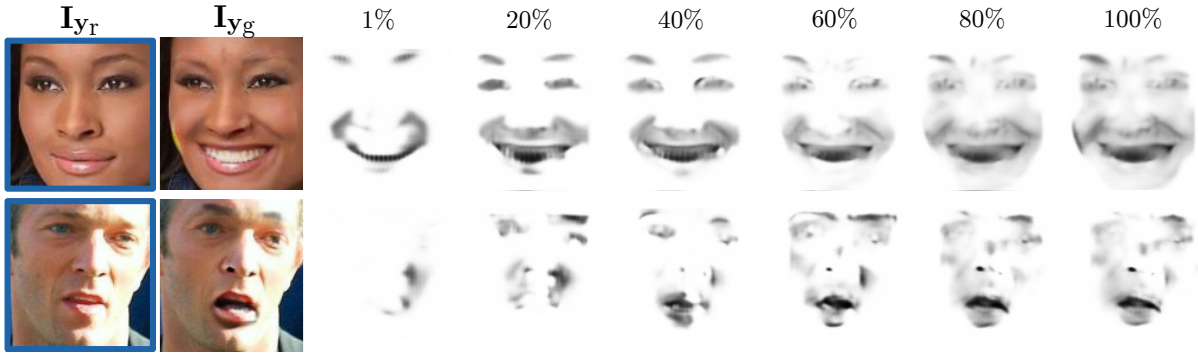


Figure 4.14: **Attention mask convergence (qualitative assessment).** Evolution of the attention mask during training. **Left to Right**: Source $\mathbf{I}_{\mathbf{y}_r}$ and generated $\mathbf{I}_{\mathbf{y}_g}$ images, respectively; and the corresponding attention mask evolution (from 1% to 100%) of the total training epochs.

by only 14 AUs. Note the large variability of anatomically feasible expressions that can be synthesized with only 14 AUs. Specially remarkable are some of the results in which parts of the face are not visible in the input image (*e.g.*, teeth) need to be hallucinated.

Comparison with the State-of-the-Art

We next compare our approach against several baselines, namely DIAT [8], CycleGAN [9], IcGAN [10] and StarGAN [11]. For a fair comparison, we consider the results of these methods trained by the most recent work, StarGAN [11], on the task of rendering discrete emotions categories (*e.g.*, happy, sad and fearful) trained and tested in the RaFD dataset [274]. Face images in this dataset are properly cropped and aligned. Since DIAT [8] and CycleGAN [9] do not allow conditioning, they were independently trained for every possible pair of source/target emotions. GANImation was also fine-tuned with the RaFD dataset. We next briefly discuss the main aspects of each approach:

DIAT [8]. Given an input image $\mathbf{I} \in X$ and a reference image $\mathbf{I}_r \in Y$, DIAT learns a GAN model to render the attributes of domain Y in the image $\mathbf{I}$ while conserving the person's identity. It is trained with the classic *adversarial loss* and a *cycle loss* $\|\mathbf{I} - G_{Y \to X}(G_{X \to Y}(\mathbf{I}))\|_1$ to preserve the

Figure 4.15: **Qualitative ablation study**. Impact of the attention mechanism and the attention loss in the generated images. **First row:** Reference expressions. **Second row:** Results using the full GANimation pipeline. **Third row:** GANimation without the attention mechanism. **Last row**: GANimation without the attention loss.

person's identity.

CycleGAN [9]. Similar to DIAT [8], CycleGAN also learns the mapping between two domains $X \rightarrow Y$ and $Y \rightarrow X$. To train the domain transfer, it uses a regularization term denoted *cycle consistency loss* that combines two cycles: $\|\mathbf{I} - G_{Y\rightarrow X}(G_{X\rightarrow Y}(\mathbf{I}))\|_1$ and $\|\mathbf{I}_r - G_{X\rightarrow Y}(G_{Y\rightarrow X}(\mathbf{I}_r))\|_1$.

IcGAN [10]. Given an input image, IcGAN uses a pre-trained encoder-decoder to encode the image into a latent representation in concatenation with an expression vector $\mathbf{y}$ to then reconstruct the original image. It can modify the expression by replacing $\mathbf{y}$ with the desired expression before passing it through the decoder.

StarGAN [11]. This approach is an extension of the *cycle loss* for simultaneously training between multiple datasets with different data domains. It uses a mask vector to ignore unspecified labels and to optimize only on known ground-truth labels. It yields more realistic results when training simultaneously with multiple datasets.

GANimation differs from these approaches in two main aspects. First, we do not condition

Figure 4.16: **Attention mask convergence (quantitative assessment).** Mean value of the attention mask over training time.

the model on discrete emotions categories, but we learn a basis of anatomically feasible warps that allows generating a continuum of expressions. Secondly, the use of the attention mask allows applying the transformation only on the cropped face, and put it back onto the original image without producing transition artifacts. As shown in Fig. 4.13, besides estimating more visually compelling images than other approaches, this results on images with higher spatial resolution.

Table 4.2 presents a quantitative analysis (that includes a user study) comparing GANimation and StarGAN [11], as a representative of current the state-of-the-art. We considered three metrics: the ACD [15], the Inception Score (IS) [16] and the user preference. ACD is the L2-distance between feature vectors of the input and generated images extracted by a face classifier[2] (the lower the better). IS is the metric used in previous approaches, that is higher for images with a large semantic content (the higher the better). For the study, we evaluated 100 randomly picked images from the RaFD dataset test set, each transformed to 5 randomly selected expressions. To compute the user preference score we asked 20 human subjects to pick the most photo-realistic generated image among 20 randomly shuffled image pairs, one generated by each method. As shown in Table 4.2 both methods have a very similar performance in terms of the quality of the generated images. ACD is slightly favorable to StarGAN and GANimation is better in IS and user preference. But recall that GANimation allows generating expressions in a continuum, while StarGAN is only able to render expressions from set of 8 emotion categories. We can conclude that GANimation retains/slightly improves the quality of StarGAN, while offering a much wider range of animation possibilities.

Method	ACD ↓	ED ↓	User Preference ↑
GANimation	0.4	**0.4**	**87%**
w/o attention	0.4	**0.4**	13%
w/o attention loss	**0.0**	4.8	0%

Table 4.3: **Quantitative ablation study.** Impact of the attention mechanism and the attention loss on the face generation results. Three metrics are considered (described in the text): *Face Distance* (ACD [15], the lower the better), *Expression Distance* (ED, the lower the better) and user preference (the higher the better).

Attention Convergence

The most critical part when training GANimation is to ensure the correct convergence of the attention mask. The fact that we are not using ground-truth supervision can easily lead to the saturation of this mask, *i.e.*, $\mathbf{A}_{H\times W} = (\mathbf{1})_{H\times W}$, meaning that $\mathbf{I}_{\mathbf{y}_o} - G(\mathbf{I}_{\mathbf{y}_o}|\mathbf{y}_f) = \mathbf{0}$, an hence the generator simply performs the identity mapping. Indeed, most terms in the loss function (see Eq. (4.15)) favor this situation, *i.e.*, if the input image is not changed (identity generator) the photo-realism, the identity preservation and the smoothness of the attention mask are maximized. To avoid this from happening, we introduced the loss term $\mathcal{L}_{\text{A}}$ that explicitly enforces regularization over the attention mask and prevents it from saturating.

Fig. 4.14 shows the convergence of the attention mask during training. We noted that in the first epochs the generator basically copies most parts of the original image (areas in white) and only introduces the basic lines that convey the new expression. After a while, the attention mask converges to a face segmentation mask that allows editing the fine details of the face such as color and shadows while leaving the original background unchanged. Fig. 4.16 shows how the amount of newly created pixels (size of the darker regions in the attention mask) increases over the training time.

Ablation Study

To further analyze the GANimation's architecture and loss components we conducted an ablation study. Performing such ablation study, however, is not trivial, as most of the model elements are crucial for convergence. D_{I} and $\mathcal{L}_{\text{I}}$ constrain the system to generate realistic images; D_{y} and $\mathcal{L}_{\text{y}}$ ensure the proper expression conditioning when generating a new sample; and $\mathcal{L}_{\text{idt}}$ enforces the model to preserve the person's identity. Removing any of these elements prevent the model from converging.

The only module that can be realistically ablated without catastrophically harming the network's performance is the attention mechanism $\mathbf{A} = G_A(\mathbf{I}_{\mathbf{y}_o}|\mathbf{y}_f)$ and its corresponding attention loss $\mathcal{L}_{\text{A}}$. Fig. 4.15 and Table 4.3 present a qualitative and quantitative ablation study of these

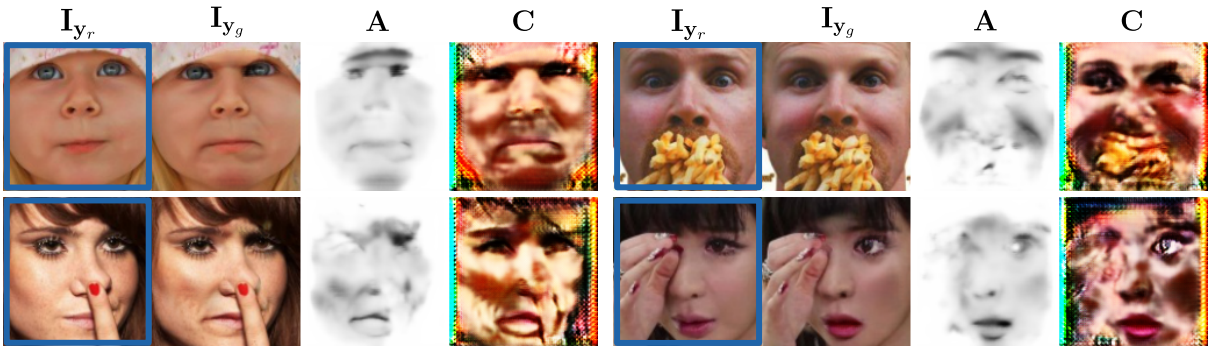


Figure 4.17: **Dealing with occlusions**. Facial editing when dealing with input images containing occlusions. In all cases, we represent (from left to right) the source image $\mathbf{I}_{\mathbf{y}_r}$; the target image $\mathbf{I}_{\mathbf{y}_g}$; the attention mask $\mathbf{A}$; and the color mask $\mathbf{C}$. **Top:** Occlusions created by external interfering objects (had and french-fries). **Bottom:** Self-occlusions created by other parts of the body (hands and hair).

two elements. For the quantitative results we considered three metrics: the ACD, the ED and the user preference. ACD is the same metric as in Subsec. 4.4.4 and ED is the l_1-distance between the generated and desired expressions (the lower the better). For the user preference we have asked 20 human subjects to pick the most photo-realistic generated image (the higher the better). For the study, we evaluated 5,000 randomly picked images from the CelebA [286] dataset, each transformed to 8 randomly selected expressions of the RaFD dataset. The model was not fine-tuned on CelebA. For the user study 20 randomly shuffled images were scored based on their photo-realism.

The quantitative results show that although we do not observe any gain on the face classification features nor on the estimated expressions, the proposed generation mechanism produces more photo-realistic images – better blended with the original background and better adjusted to the scene illumination. This is clearly reflected by the user study. When no attention is used, the cropped face bounding boxes are visible in the generated image and the illumination is not consistent (see Fig. 4.15-*w/o attention*). By contrasts, when using the proposed generator the background is perfectly blended and the illumination of the background and the generated image are consistent (see Fig. 4.15-*GANimation*).

The ablation study also demonstrates the necessity of introducing the proposed attention loss $\mathcal{L}_{\mathrm{A}}$ for the proper convergence of the model (see Table 4.3). When removing it, the obtained ACD metric is 0.0, meaning $\mathbf{I}_{\mathbf{y}_r} = \mathbf{I}_{\mathbf{y}_g}$, that is, the output image is identical to the input image.

Dealing with Occlusions

We next explicitly evaluate the robustness of the proposed approach to partial occlusions of the

input face. The results are shown in Fig. 4.17. Interestingly, the attention mask tags the occluded pixels in white, meaning that these pixels will not be changed by the generator when creating the new expression. This is another interesting property of the attention mechanism, which besides learning a smooth foreground-background blending function, it also learns to ignore the static elements of the image that do not participate in the generation of the facial expression, like hats, glasses, hands or interfering objects. Recall that this is learned in a weakly supervised manner.

Images in the Wild

As previously seen, the attention mechanism not only learns to focus on specific areas of the face but also allows smoothly merging the original and the generated image background. This allows our approach to be easily applied to images in the wild while still maintaining the resolution of the original images. For these images we follow the detection and cropping scheme we described before. Fig. 4.18 shows two examples on these challenging images: the first example illustrates our model's performance on a multiple-face editing task with complex illumination; the second example deals with a non-human-like facial skin texture distribution, which is obviously not observed at training time. Note how the attention mask allows for a smooth and unnoticeable merging between the entire frame and the generated faces.

Pushing the Limits of the Model

We next push the limits of our network and discuss the model limitations when dealing with extreme situations such as stone-like skin, drawings and face sketch abstractions. We have split success cases into six categories which we summarize in Fig. 4.19-top. The first two examples (top-row) correspond to human-like sculptures and non-realistic drawings. In both cases, the generator is able to maintain the artistic effects of the original image. Also, note how the attention mask ignores artifacts such as the pixels occluded by the glasses. The third example (second-row, left) shows robustness to non-homogeneous textures over the face. Observe that the model is not trying to homogenize the texture by adding/removing the beard's hair. The second-row, right example, corresponds to an anthropomorphic face with non-real texture. As for the Avatar image, the network is able to warp the face without affecting its texture. The next category (third-row, left) is related to non-standard illuminations/colors for which the model has already been shown robust in Fig. 4.7. The last and most surprising category is face-sketches (third-row, right). Although the generated face suffers from some artifacts, it is impressive how GANimation is still capable of finding sufficient features on the face to transform its expression from worried to excited.

Figure 4.18: **Qualitative evaluation on images in the wild. Top:** We represent an image (left) from the film "*Pirates of the Caribbean*" and an its generated image obtained by our approach (right). **Bottom:** In a similar manner, we use an image frame (left) from the series "*Game of Thrones*" to synthesize five new images with different expressions.

The fourth and fifth rows of Fig. 4.19 show a number of failure cases. The first case is related to errors in the attention mechanism when given extreme input expressions. The attention does not sufficiently weight the color transformation causing transparencies. The second case (fifth row, right) shows failures with non-previously seen occlusions such as an eye patch causing artifacts in the missing face attributes. The model also fails when dealing with non-human anthropomorphic distributions as in the case of cyclopes. Also, in this case, the face detection failed to detect the Cyclopes face forcing the generator to directly modify the original image without previously cropping the face. Lastly, we tested the model behavior when dealing with animals and observed artifacts like human face features.

4.5 Unsupervised Image-to-Video Cloth Transfer

The last years have witnessed a surge in e-commerce and, in particular, online fashion shopping. Since 2003 fashion retail sales have experienced a steady annual growth [287] reaching 19.8% of the total fashion retail sales in the United States in 2017 [288]. However, most users prefer in-store shopping in order to get the proper fit for clothing, see how items combine with other garments and how would they all look on them. In the near future, we expect that people will be able to easily extract and virtually try on cloth from any image on the web without the need for specialized hardware. In this section, we take a step forward towards this future by leveraging all the knowledge on image editing acquired in the previous section to edit images. But now,

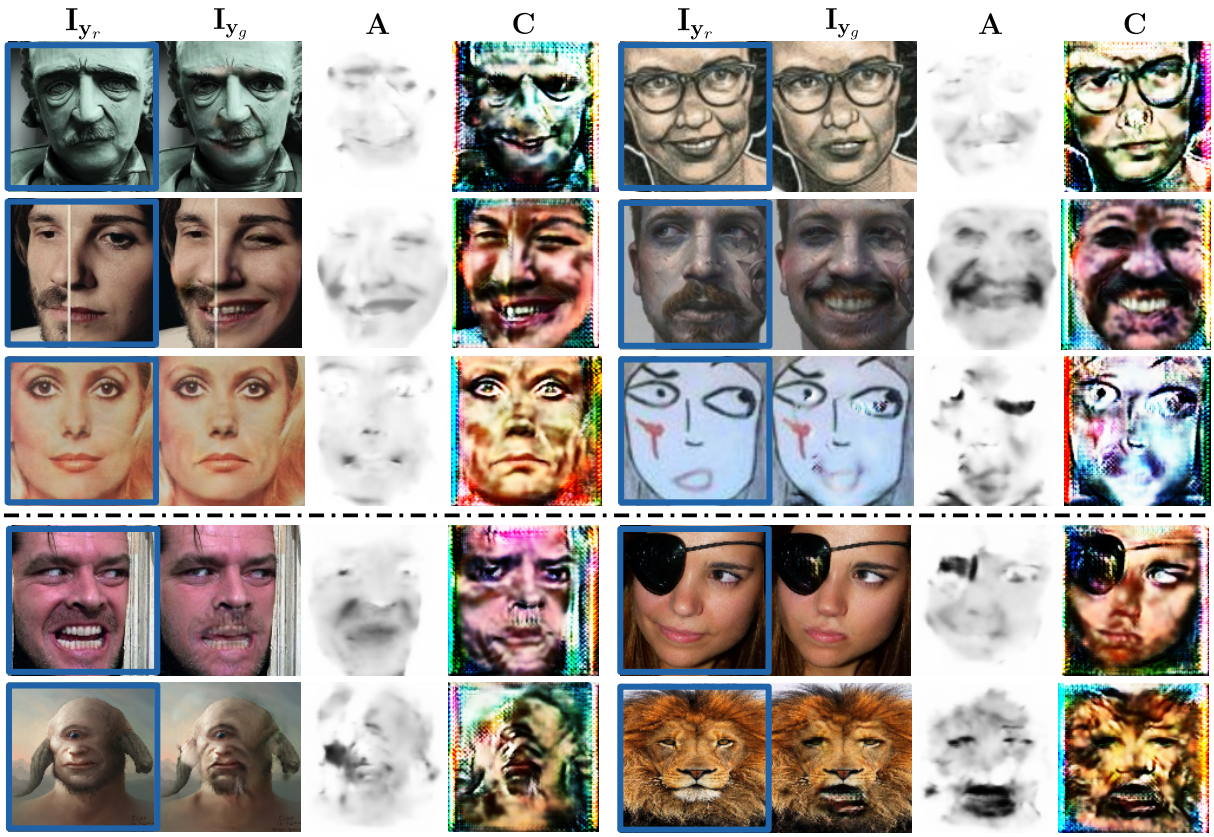


Figure 4.19: **Success and failure cases.** In all cases, we represent the source image $\mathbf{I}_{\mathbf{y}_r}$, the target image $\mathbf{I}_{\mathbf{y}_g}$, the attention mask $\mathbf{A}$ and the color mask $\mathbf{C}$. **Top:** Some success cases in extreme situations. **Bottom:** Several failure cases.

instead of editing face expressions we will be editing cloth.

Fig. 4.20[3] illustrates the main problem addressed in this section. In the left image, we observe a person with an iconic computer science clothing and in the right image the result of our algorithm to automatically transfer the clothing style to other subjects under different poses, backgrounds, lightnings and body shapes. Our model is able to synthesize space-time consistent novel views of the source clothing, while simultaneously fitting them to the target person body shape and maintaining the original background. The proposed method is learned in an unsupervised fashion, that is, we do not require pairs of images of the same person with different clothing or poses, or the same person with same clothes in different positions. Learning the models without this supervision, allows training with large amounts of data which strengthens the robustness of our method to changes in body pose, background, occlusions and clothing.

[3]In this section faces have been blurred out to preserve the persons' private identity.

Figure 4.20: **Example of visual clothing transferring.** Left, original image of an iconic computer science clothing style. Right, the result of transferring the cloths from the left image to other images containing one or several subjects in unconstrained poses and background. This figure illustrates results in individual images, but the system is able generate space-time consistent novel views of clothing in videos. *Note: All faces in this section have been blurred out to preserve the persons' private identity.*

To address all above-mentioned challenges, we combine a clothing segmentation output with a temporally-consistent Generative Adversarial Network. Our main contribution consists in equipping a standard GAN architecture with a memory module that progressively refines a source texture map and adapts it to the target person, by filling occluded regions and adapting to new lighting conditions and body pose. This section is related to recently proposed deep-learning approaches for transferring clothes [248, 249]; however, while these models provide visually compelling results, they typically rely on 3D human models, and their results are limited to non-cluttered backgrounds, mild lighting conditions and require supervised training. Our TC-GAN offers a simple but effective *unsupervised* image2Video approach that is shown to be robust results across pose, background, lighting and body variability without the need of knowing the underlying geometry of the body nor the physics ruling the cloth deformations.

In summary, this section main contributions are: 1) the first approach to tackle the clothing transfer problem in a non-supervised manner. This allows training directly on images randomly downloaded from the internet, while other methods require specialized datasets; and 2) previous video generation methods tackled temporal coherence through motion compensation, latent representations and specialized losses. Instead, we propose a physical memory that stores

clothing texture across frames and guarantees temporal consistency when rendering images in long video sequences.

4.5.1 Problem Formulation

Let $\mathbf{I}_c \in \mathbb{R}^{H \times W \times 3}$ be an input RGB image of a dressed person (source), and let $\mathbf{x}_1^T = (\mathbf{X}_1, \ldots, \mathbf{X}_T)$ be the target video, where $\mathbf{X}_t \in \mathbb{R}^{H \times W \times 3}$ and the subindex t denotes the video frame. The target video can be of the same person in $\mathbf{I}_c$ or a different one. Our goal is to learn a mapping $\mathcal{M}$ to transform $\mathbf{x}_1^T$ into an output video $\mathbf{y}_1^T$ where target person is realistically dressed with the clothes of $\mathbf{I}_c$. That is, we aim to transfer cloth items from the source image $\mathbf{I}_c$ to the target video $\mathbf{x}_1^T$ by learning the mapping $\mathcal{M} : (\mathbf{x}_1^T, \mathbf{I}_c) \rightarrow \mathbf{y}_1^T = (\mathbf{Y}_1, \ldots, \mathbf{Y}_T)$.

One of our major contributions is that we propose learning $\mathcal{M}$ in an unsupervised manner, that is, we do not require pairs of images of the same person under different clothes or poses, or the same person wearing the same clothes under varying body postures. Instead, our training data consists merely of N input RGB images $\{\mathbf{I}_c^{(n)}\}_{n=1}^N$.

The output video $\mathbf{y}_1^T$ is expected to be meet the following criteria: (i) the transferred cloth items must be adjusted to the pose and body shape of the target person; (ii) hallucinated views of the source clothes that are not visible in the source image $\mathbf{I}_c$, must be photo-realistic and consistent with the visible parts; (iii) transferred cloth items must be consistent with the illumination of the target video $\mathbf{x}_1^T$; (iv) target elements as body parts, background and non-transferred cloth items must remain fixed; and (v) the texture across the output frames of $\mathbf{y}_1^T$ must be consistent in space and time.

4.5.2 Time Consistent Memory GAN

In this subsection we describe the main contribution of this work, namely the time-consistent GAN to photo-realistically transfer clothing between images of dressed people. The overall architecture is depicted in Fig. 4.21. Before going deeper into the details of this network, let us summarize its general functioning.

At test time, the source image $\mathbf{I}_c$ is used to initialize the state of the texture map $\mathbf{T}_0$ to be transferred. Since this texture map is estimated from a single view, it will contain many regions with missing information. Each target frame $\mathbf{X}_t \; \forall t \in \{1, ..., T\}$ is then transformed to the output frame $\mathbf{Y}_t$ following a two step process in which first the color information of the target cloth regions are queried from $\mathbf{T}_{t-1}$. This results in intermediate images $\mathbf{X}'_t$ that contain a number of holes (yellow regions in Fig. 4.21). In the second step, the generator fills these holes and renders the output images $\mathbf{Y}_t$. Note that by introducing the texture memory, the generator only

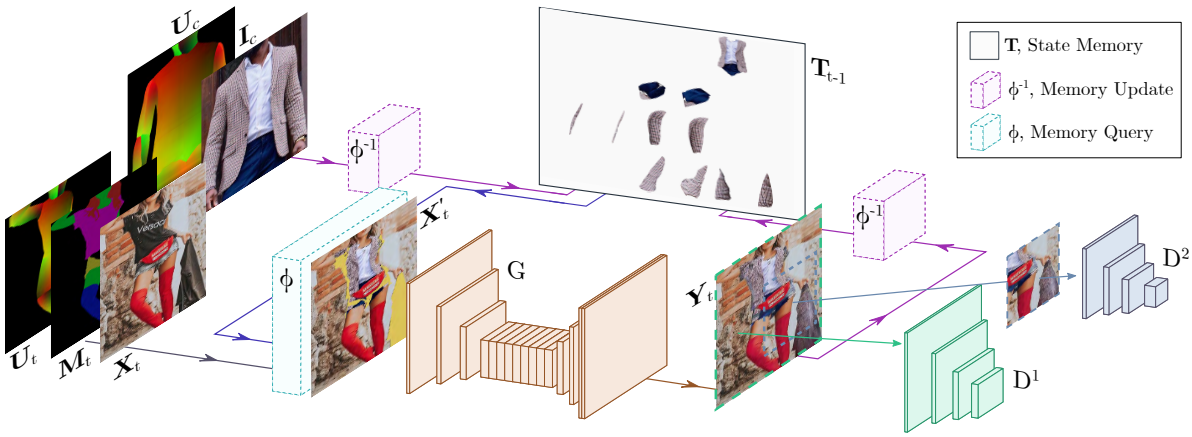


Figure 4.21: **Overview of our approach for image-to-video cloth transfer.** The proposed architecture consists of three main blocks: a generator G to transfer cloth items; a memory $\mathbf{T}$ that stores textures across long video sequences; and a multiscale discriminator D to evaluate the photo-realism of the generated image and its consistency with the cloth segmentation labels. Note that our system does not require supervision, *i.e.*, no pairs of images nor videos of different persons under the same clothing.

needs to perform a texture inpainting task, instead of having to learn the entire cloth mapping as in previous works [25, 249, 289]. The inpainted regions are cumulatively added to the texture memory that shall be used in the subsequent frames. By doing this, we guarantee temporal consistency. Finally, a multilevel PatchGan [52] discriminator $D(\mathbf{Y}_t, \mathbf{M}_t)$ evaluates the photo-realism of the generated frame and its compatibility with the input cloth segmentation labels $\mathbf{M}_t$.

Dataset Pre-processing - Cloth Segmentation

In order to learn the cloth mapping transformation, we automatically enrich the input dataset of RGB images with segmentation masks $\{\mathbf{M}^n\}_{n=1}^N$ computed for each input image $\mathbf{I}_c^n$, where $\mathbf{M}^n \in \mathbb{R}^{H \times W \times S}$ are the segmentation masks for S cloth labels. We further augment the dataset with random occlusions, rotations, translations and color jitter. Specifically, occlusions are randomly introduced over the cloth and body regions (excluding the face) to simulate non-visible parts of the source clothing to be hallucinated. To help improving the blending of the transferred cloth onto the image background we also add occlusions on the cloth-background boundaries. Fig. 4.22 shows the type of input information we will use later to learn the mapping $\mathcal{M}$. We next describe the network we use for performing cloth segmentation. Note that this network is independently trained from the one to perform the actual cloth mapping, described in Sec. 4.5.2, and which is the main focus of this work.

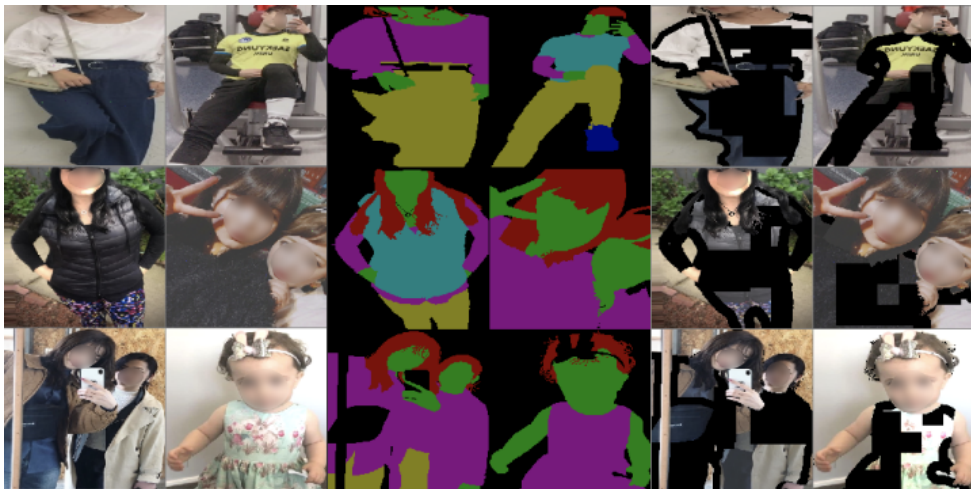


Figure 4.22: **Dataset examples.** The dataset used to learn the cloth mapping is originally made of only RGB images (left). We automatically enrich it by computing cloth segmentation masks (center) and artificially introducing occlusions (right). We also introduce color and brightness perturbations, which is shown in some of the left-most images.

Existing state-of-the-art cloth segmentation algorithms [7, 290, 291] are intended for fine-grained situations with a large variety of cloth items. In our case, segmentation labels will be used to guide the cloth transfer process between people under potentially different outfits. Therefore, we define a reduced number of high-level categories that can be shared even under different clothing styles. Concretely, the segmentation labels we consider are: hair, skin, top-layer1, top-layer2, bottom and shoes. For estimating such segmentation labels we built our own architecture, which we describe next.

Cloth segmentation is performed using a PSPNet architecture [292] with a Resnet50 backbone, initialized with pre-trained Imagenet weights. After the feature map, we introduce a pyramid pooling module with pooled feature maps of bin sizes $\{1 \times 1, 2 \times 2, 3 \times 3, 6 \times 6\}$ with average pooling. The model is trained with 69K pairs of image - cloth labels of size 512×512. We used SGD with learning rate of 0.01. Momentum and weight decay are set to 0.9 and 0.0004 respectively. We set batch size to 32 in training. We train for 100 epochs on 8 GPUs.

This initial architecture, however, lacks of any strategy for enforcing temporal consistency. For this purpose we extended the *single-image segmentation* network with a tracking methodology, by training a second network which is fed with an RGB image and the segmentation mask predicted in the previous frame. To avoid having to label video frames, this tracking model is trained using as input a randomly warped version of the RGB image and the *unwarped* segmentation mask detected by the single-image segmentation module. Note that by doing

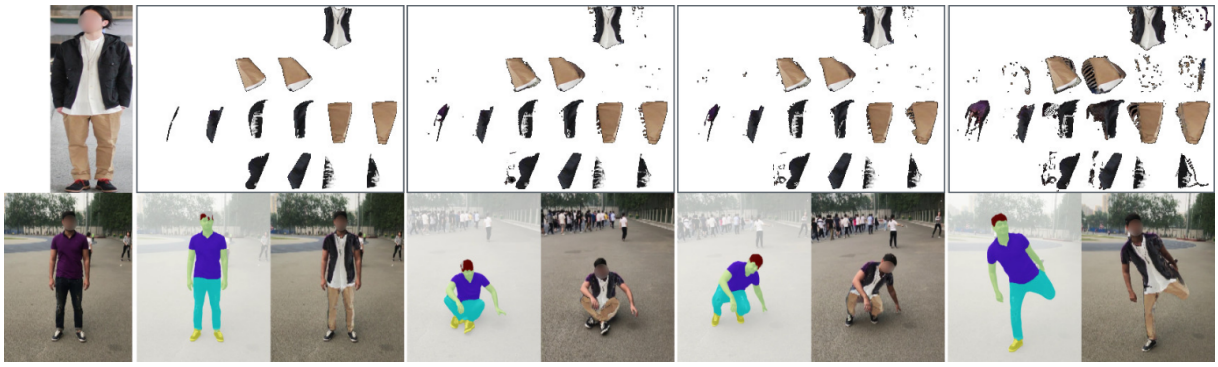


Figure 4.23: **Memory state and segmentation tracking. Top-Left:** Input source image $\mathbf{I}_c$. **Bottom-Left:** First frame $\mathbf{X}_t$ of the target video in which we seek to transfer the clothes of $\mathbf{I}_c$. **Top-row (columns 2-5)**: Visualization of the memory $\mathbf{T}_t$, initially containing only the parts visible in $\mathbf{I}_c$. Novel regions hallucinated for the frames $\mathbf{X}_t$ are progressively added into the texture map. **Bottom-row (columns 2-5)**: Segmentation masks $\mathbf{M}_t$ (automatically estimated) and cloth transfer results $\mathbf{Y}_t$.

this we are coupling the single frame and the tracking models in a way that the tracking module refines the previous frame estimation. This allowed obtaining very consistent temporal segmentation results.

At inference, the single-frame model runs first on an RGB input image and predicts a segmentation mask for that frame. The tracking model is then applied over the subsequent frames by taking as input the mask from the previous frame plus the RGB input of the current frame. To prevent drifting we designed a simple re-initialization mechanism in which the tracking model runs until the *intersection over union* (IoU) of the current frame mask and the previous frame mask is above a threshold λ_{IoU}. When it drops below λ_{IoU} the single-frame module is run again. λ_{IoU} is set to 0.85. In the experimental subsection we show several results of this approach.

Model Architecture

We next describe in detail each of the main components of our network.

Texture Memory: We define the memory as the estimated texture map of the target cloth. We use the same body partition and UV parametrization as in DensePose [251]. At $t = 0$ the memory is initialized with the cloth's visible parts from the source image $\mathbf{I}_c$. At each time step, the cloth regions not seen in the target image are hallucinated by the generator and cumulatively added to the state memory. An example of how the memory evolves across a video sequence is shown in the top of Fig. 4.23.

Memory Query: The texture memory can be accessed by both the discriminator an the generator. In the generation phase, the memory is queried using the mapping $\Phi : (\mathbf{X}_t, \mathbf{U}_t, \mathbf{M}_t, \mathbf{T}_{t-1}) \rightarrow$

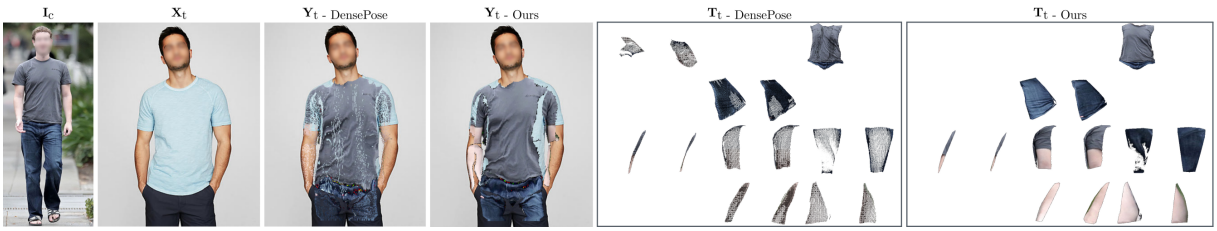


Figure 4.24: **From the texture map to the image and viceversa. Top:** When mapping from the texture map to the image naively using DensePose correspondences several artifacts appear (left). We partially remove them using a Piecewise Affine Transformation (PAT) strategy (right). **Bottom:** Similar noisy patterns appear when updating the texture map with the image information (left). Again we apply the PAT to smooth the mapping.

$\mathbf{X}'_t$, which renders the source cloth into the target frame $\mathbf{X}_t$. In order to perform this mapping, we first extract dense 2D correspondences $\mathbf{U}_t$ between the input the image $\mathbf{X}_t$ and a 3D body model, which implicitly provides the mapping onto the texture map. The correspondences are obtained with the pretrained DensePose network [251]. Yet, when using directly DensePose correspondences to assign a texture color to every image pixel a number of artifacts arise in the form of irregular stripe lines with missing information (see Fig. 4.24-c). We remove these by smoothing the texture-to-image map using piecewise affine transformation (see Fig. 4.24-c). While this operation improves the mapping, the rendered image $\mathbf{X}'_t$ still is prone to contain several regions with incomplete information.

Cloth Segmentation: The segmentation mask $\mathbf{M}_k$ for each video frame is inferred using the network we described in Sec. 4.5.2. This network performs the mapping $\Omega : (\mathbf{X}_k, \mathbf{M}_{k-1}) \rightarrow \mathbf{M}_k$. Note that, as we mentioned above, the segmentation network is fed with the input RGB image and the mask $\mathbf{M}_{k-1}$ at the previous time step. This helps to improve the temporal consistency of the segmentation, heavily reducing the jittering in the boundaries of the segmented regions.

Generator: The incomplete image $\mathbf{X}'_t$ and the input segmentation masks are passed to the generator $G : (\mathbf{X}'_t, \mathbf{M}_t) \rightarrow \mathbf{Y}_t$. We force G to primarily focus on the segmented regions of the body, by adapting the lighting in the regions of $\mathbf{X}'_t$ which already have texture information, and inpainting those which do not have, corresponding to the areas that were not visible in the previous frames $\mathbf{X}_1, \ldots, \mathbf{X}_{t-1}$.

Following the recent works in Self-Attention Generative Adversarial Networks [293–295], we implemented a mechanism in the generator that allows the feature maps to be built using relationships between widely separated spatial regions. The intuition behind this design choice is that inpainting cloth textures is a task in which not only local information needs to be considered, but also the details of other parts of the clothing that can be relatively far apart (*e.g.*shirt sleeves, shoes).

Memory Update: The new regions in $\mathbf{Y}_t$ the generator has hallucinated are mapped back to the texture memory using the inverse of the mapping we considered during the memory query phase, that is, $\Phi^{-1} : (\mathbf{Y}_t, \mathbf{M}_t, \mathbf{U}_t) \rightarrow \mathbf{T}_t$. Again, the 2D correspondences $\mathbf{U}_t$ are estimated using DensePose, producing noisy stripe line patterns into the texture map (see Fig. 4.24-c). The piecewise affine tranform is used one more time to alleviate this problem (see Fig. 4.24-e).

Note that we only update the texture map with the new regions the generator hallucitaded in the current frame $\mathbf{X}'_t$. The texture regions already observed in previous frames are not modified. This strategy was the one that gave us better results in terms of temporal consistency of the textures mapped onto the images rendered with the new clothes. We considered other updating rules (*e.g.*a weighted update of previous areas with temporal decay), but did not perform that well.

Multilevel Discriminator: The photo-realism of the generated image $\mathbf{Y}_t$ is evaluated with the network $D(\mathbf{Y}_t, \mathbf{M}_t)$. Its structure is similar to the multilevel PatchGan [229], which is made of two discriminators with identical architecture that operate at different image resolutions, one having a global view of the image to guide the generator to produce cloth labels, and the other focused on fine texture details. Each discriminator computes a probability map for overlapping image patches, that assesses the level of photo-realism of the rendered image and the consistency with the input cloth segmentation labels.

4.5.3 Learning the Model

After having automatically segmented the N input RGB images $\{\mathbf{I}^n\}_{n=1}^N$, we can redefine our training set as N tuples $\{\mathbf{I}^n, \mathbf{M}^n\}$ of unordered images and their corresponding segmentation masks. Recall that we do not consider neither videos nor ground truth output images. Despite not having direct supervision, we carry on simple but effective data augmentation (as described in Sec. 4.5.2) and define problem specific loss functions that allow our model to learn high-quality cloth transfers maps.

Concretely we train our model with a loss function made of three terms: an *image adversarial loss* to push the distribution of the generated images to the distribution of the training images; a *masked perceptual loss* to stabilize the training loss by penalizing high-dimensional features in the generated image; and a *feature matching loss* to further stabilize the convergence by forcing the generator to match statistics at multiple feature levels of the discriminator. We next describe each of these components:

Image Adversarial Loss: In order to optimize the parameters of the generator G and learn the distribution of the training data, we extend the commonly used min-max strategy game between the generator and the discriminator proposed by Goodfellow *et al.* [40]. To enforce the model

not just to produce photo-realistic images but also to be consistent with the cloth segmentation labels, we add an extra term in the adversarial loss that aims to classify a mismatched image-mask pair as a negative sample. Formally, let $\mathbf{X}$ be the input image with its corresponding cloth segmentation labels $\mathbf{M}$; $\mathbb{P}_r$ the data distribution of the input images and $\mathbb{P}_g$ the distribution of the generated images $\hat{\mathbf{X}} = G(\mathbf{X}', \mathbf{M})$; and $\hat{\mathbf{M}}$ a segmentation mask randomly chosen from the training set. We then define the extended adversarial loss $\mathcal{L}_I$ is defined as follows:

$$\begin{aligned}\mathcal{L}_I = \mathbb{E}_{\mathbf{X}\sim\mathbb{P}_r}[\log(D(\mathbf{X}, \mathbf{M}))] + \lambda(\mathbb{E}_{\mathbf{X}\sim\mathbb{P}_r}[\log(1 - D(\mathbf{X}, \hat{\mathbf{M}}))] \\ + \mathbb{E}_{\hat{\mathbf{X}}\sim\mathbb{P}_g}[\log(1 - D(\hat{\mathbf{X}}, \mathbf{M}))]),\end{aligned} \tag{4.17}$$

where $\lambda = 0.5$ is a penalty coefficient to balance the positive-negative rate.

Masked Perceptual Loss: In order to stabilize the training, we added a perceptual loss [268] masked over the clothing regions. This loss penalizes the L_1 distance between the original and inpainted images after being projected into a high dimensional feature space. It can be written as:

$$\mathcal{L}_P = \sum_{n=1}^{N} \left\| \mathbf{M} \odot (\Psi_n(\mathbf{X}) - \Psi_n(\hat{\mathbf{X}})) \right\|_1, \tag{4.18}$$

where $\odot$ is the pixelwise multiplication and Ψ_n the activation maps of the nth layer of a pre-trained VGG-16 network [270].

Feature Matching Loss: To further stabilize the training process we penalize high level features on the discriminators [229]. Similar to what was done in the loss $\mathcal{L}_P$, the generator is enforced to match statistics of the original and inpainted images at multiple feature levels of the two discriminators described in Sec. 4.5.2. If we denote by D_l^k the l-th layer of the k-th discriminator ($k = \{1, 2\}$), the loss $\mathcal{L}_F$ is defined as:

$$\mathbb{E}_{\mathbf{X}\sim\mathbb{P}_r, \hat{\mathbf{X}}\sim\mathbb{P}_g} \sum_{k=1}^{K} \sum_{l=1}^{L} \frac{1}{N_l^k} \left\| D_l^k(\mathbf{X}, \mathbf{M}) - D_l^k(\hat{\mathbf{X}}, \mathbf{M}) \right\|_1, \tag{4.19}$$

where N_l^k is a weight regularizer denoting the number of elements in the l-th layer of the k-th discriminator.

Total Loss: The final min-max problem that combines all previous losses is:

$$G^\star = \arg\min_G \max_D \lambda_I \mathcal{L}_I + \lambda_P \mathcal{L}_P + \lambda_F \mathcal{L}_F \tag{4.20}$$

where λ_I, λ_P and λ_F are the hyper-parameters that control the relative importance of every loss

term and $G^\star$ draws samples from the data distribution.

Implementation Details

The generator consists of a U-Net architecture [230] with 4 feature levels. We have extended the bottleneck of this network with 7 ResNet blocks and two self-attention layers, introduced after the 3th and 6th ResNet blocks. Self-attention layers are known to be very memory demanding. As suggested in [294], we have reduced these requirements by subsampling the feature maps using maxpooling.

For discriminator networks, we use PatchGan [52], extended to multi-level as in [229]. The two discriminators have the same architecture, each is formed by 4 down-convolutions with InstanceNorm and a Leaky ReLU with slope 0.2. After the last layer, we apply a convolution to produce a 1 dimensional output. The first convolution layer has no normalization.

The model is trained on 69.000 pairs of image - cloth labels of size 256×256. We use Adam [38] with beta1 0.5, beta2 0.999, learning rate of 0.0001, and batch size 40 for both the generator and discriminators. We train for 60 epochs. The weight coefficients for the loss terms in Eq. (4.20) are set to $\lambda_I = 1$, $\lambda_P = 0.05$ and $\lambda_F = 10$.

4.5.4 Experimental Evaluation

We next report a thorough evaluation of our algorithm, including qualitative results on still images and video sequences and a quantitative assessment using the IS metric. In all results we do not assume ground truth cloth segmentation. We use the segmentation masks automatically computed using the network we have described in Sec. 4.5.2.

Images in the Wild

We start evaluating the ability of our approach to transfer clothing between images in the wild, with varying cloth items, unconstrained body poses and cluttered backgrounds. Fig. 4.25 shows the results in four examples organized as follows: the top row are the reference (source) images $\mathbf{I}_c$ and the left-most column are the target images $\mathbf{X}_t$ in which we seek to transfer the source clothes. For every example, we represent the results of the two estimation steps: (i) memory query $\Phi : (\mathbf{X}_t, \mathbf{T}_{t-1}) \rightarrow \mathbf{X}'_t$ where $\mathbf{X}'_t$ is a first estimation obtained with a 2D piecewise affinity transformation of the texture. This initial estimation still contains missing parts, highlighted in yellow; (ii) Final transferred image with cloth completion $G : (\mathbf{X}'_t, \mathbf{M}_t) \rightarrow \mathbf{Y}_t$.

The results demonstrate the ability of our model to transfer very different types of upper-body items, from short to long sleeves, t-shirts, jackets, jacket suits and long coats. For example, observe the columns 4 and 5 corresponding to the second reference image. Our model correctly

Figure 4.25: **Cloth transfer.** Results from transferring the reference cloths ($\mathbf{I}_c$) in the top row to the target images in the first column ($\mathbf{X}_t$). For every transfer, we report the initial estimation $\mathbf{X}'_t$ and the final result $\mathbf{Y}_t$. Missing areas after removing the original cloth and warping the reference cloth are marked in yellow. These are the areas the generator is trained to inpaint.

introduces the long coat around the clothes of the original target images. Similarly, the learned model is also robust to different types of lower-body clothing. Interestingly, note that it is also effective to transfer the clothes between images of the body in which some of the reference parts are largely occluded. For instance, as shown in the right-most column, the partially observed source trousers are correctly inpaited on the complete trouser area of the target woman in row #2. The model also shows robustness to disconnected parts, such us the case of the second woman (row #3) holding a jacket.

The examples in Fig. 4.25 correspond to still target images. In Fig. 4.23 we already showed an example of how clothes were transferred into a video sequence.

Method	Mean	Std
Pose GAN [25]	2.46	0.80
Pose Variational U-NET [296]	2.79	0.36
VITON [248]	3.11	0.68
Ours $\mathbf{X}'_t$ (only Memory Query)	3.47	0.56
Ours $\mathbf{Y}_t$ (Memory Query + Generator Completion)	**3.94**	**0.89**
Real Data (Upper Bound)	4.21	0.62

Table 4.4: **Quantitative evaluation**. Evaluation using the IS metric (the highest the better).

Quantitative Evaluation

We next provide a quantitatively evaluation of our algorithm, compare it against the state-of-the-art methods and perform an ablation study. The assessment will be done using the IS [16], a popular metric for judging the image outputs of GAN and which focuses on the overall quality of the generated images rather than on the pixel-level accuracy. The IS takes a list of images and returns a single floating point number, the score, which will be higher for images with a large semantic content.

Table 4.4 reports a quantitative comparison with the state-of-the-art approaches [25] and [296]. These works allow rendering novel views of clothed people but cannot change the identity of the person (*i.e.*, the source and target individuals are the same). We also consider [248], that given a photo of a cloth item, can transfer it into a person while retaining his/her pose. It is worth to point that [249, 289] are also closely related works, but they do not provide code nor benchmark results, preventing being directly compared.

Additionally, Table 4.4 provides an ablation study of our approach, and compares the intermediate results $\mathbf{X}'_t$ we obtain right after querying the memory and performing the piecewise affine transformation, with the final results $\mathbf{Y}_t$ obtained after applying the learned generator. To give significance and a reference basis to the results, we also included the IS of the real training images, which would act as an upper bound.

The IS results on the table reveal that our model outperforms the previous approaches by a significant margin. Even the intermediate results obtained by *memory querying* are a very reliable approximation, achieving high-quality results that are further refined after completing the missing regions with the generator.

Mapping Complex Clothing

We next evaluate the performance of our model under complex mappings between the source and target outfits, such as the trouser-to-skirt mappings shown in Fig. 4.26. It is in this type of scenarios where the high-level segmentation we propose becomes very relevant, as *e.g.*both

Figure 4.26: **Complex cloth mapping.** Our approach can handle situations where the source and target clothes have vary different topology (*e.g.*we can robustly transfer trouser textures to long skirts and dresses.

'trousers' and 'skirts' will be assigned a 'bottom' label, and our model has learned mappings between source-to-target clothes that have the same segmentation label. In any event, when there exist such large geometric differences between the source and target clothes, the initial estimation $\mathbf{X}'_t$ is fairly inaccurate, which large incomplete regions, and requires from the action of the generator to fill these areas. Nevertheless, as shown in the figure, the results are still very photo-realistic.

Multi-Person Cloth Transfer

Another situation we have evaluated is when several people appear in the target image (see Fig. 4.27). To increase the inference speed in these situations, we have designed our network such that it can generate as many cloth instances as necessary. This allows performing single-shot multiple instance transfer, that is, that all clothes of all subjects in the image are simultaneously transferred in a single pass, instead of doing one cloth item at a time.

The quality of the results in images with two people (we have no larger number of individuals per image in our dataset) is on a par with the single person images. One interesting result is

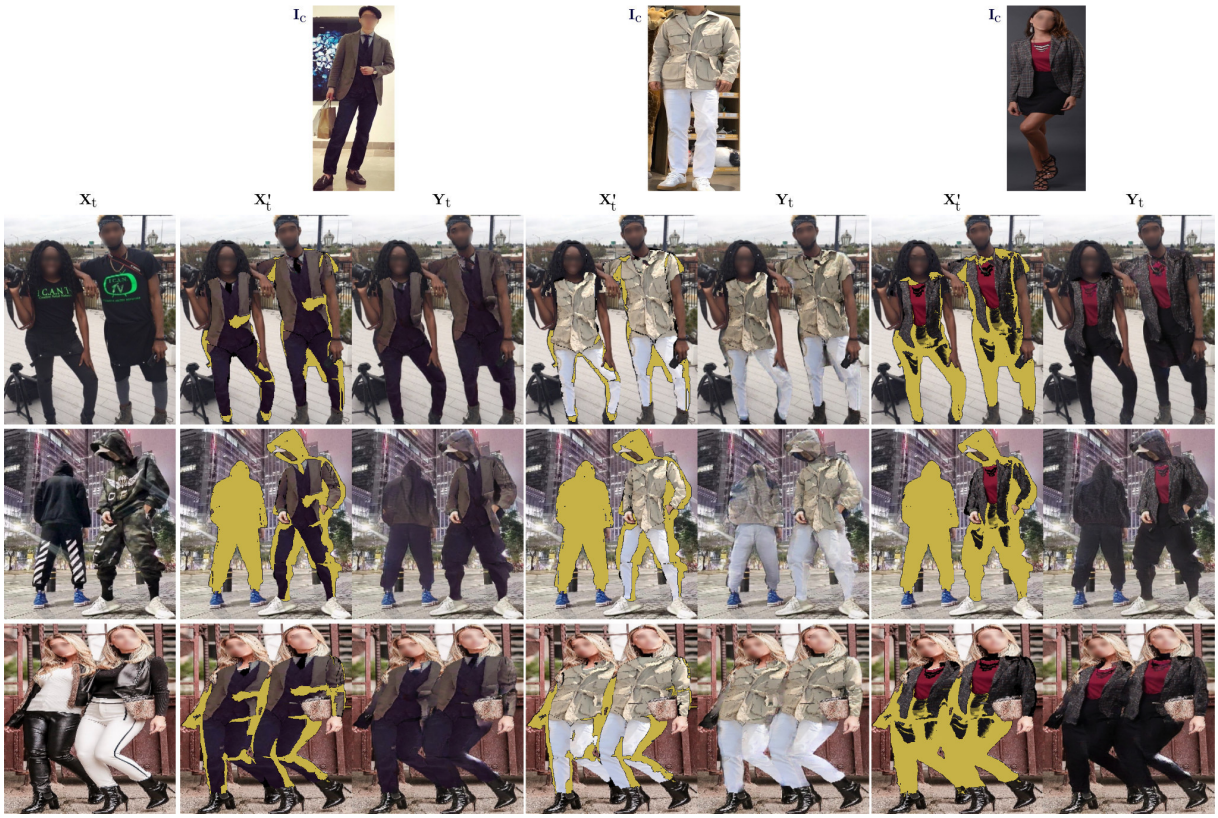


Figure 4.27: **Multi-Person cloth transfer.** Results of our cloth transfer algorithm when dealing with multiple people in the target image.

shown in the second target image (row #3), in which one person is facing back. In this case, no texture from the reference image can be initially copied into the memory (as the source and target image have no common visible regions). The full mapping is then performed by the generator, that needs to hallucitate the texture of all person clothes. The results are still very satisfactory in this case.

Videos in the Wild

We next present additional results (complementing Fig. 4.25) on image-to-video cloth transfer. Some frames of each video sequence are shown in Figs. 4.29, 4.30, 4.31. In each sequence the figure is organized as follows: the left-most column corresponds to three reference cloth images $\mathbf{I}_c$ (source) to be transferred to the target images $\mathbf{X}_t$ displayed on the top row. For every video frame, we show the cloth segmentation estimation $\mathbf{M}_t$ and the output images $\mathbf{Y}_t$ with the transferred clothes.

Note that our model shows remarkable temporally consistent results and robustness to cluttered backgrounds, different body postures (Figs. 4.29, 4.30 and rotations (Fig. 4.31). Fur-

Figure 4.28: **Failure cases.** The image displays several failure cases of our system, due to segmentation errors (top), unrealistic hair inpainting (middle) and incorrect texture warping (bottom). See text for details.

thermore, in contrast to previous methods [247, 248], we do not require the person nor the reference cloth to be initialized from a predefined position. This provides our system with a high flexibility towards being applied on unrestricted images from the Internet.

Pushing the Limits of the Model

We next push the limits of our model and discuss its limitations and failure cases. We have split the errors into three categories, as shown in Fig. 4.28. The first failure case is produced by errors in the segmentation, causing $\mathbf{X}_t$ to keep part of the original cloth texture that is later propagated by the generator across the rendered image. This is particularly clear in the the white regions of the skirt, in Fig. 4.28-top. The second source of error is produced when inpainting hair regions, specially when they cover large parts of the image (Fig. 4.28-middle). Yet, realistically rendering hair is an open research field, out of the scope of this work, but which we plan to tackle in the future. Finally, since we do not estimate the underlying geometry, there are situations where the propagation of the texture is not consistent.

4.6 Summary

In this chapter we have presented three novel generative methods to model the nonrigid articulated human body. In experimentation we demonstrated these methods to be capable of sampling novel photo-realistic views of a person, face expressions and cloth in the wild.

For generating new images of a person under arbitrary poses we have presented a GAN model that can be trained in a fully unsupervised manner. These advances state-of-the-art, which so far, had only addressed the problem using supervision. To tackle this challenge, we have proposed a new framework that circumvents the need of training data by optimizing a loss function that only depends on the input image and the rendered one, and aims at retaining the style and semantic content of the original image. Quantitative and qualitative evaluation on the DeepFashion dataset shows very promising results, even for new body poses that highly differ

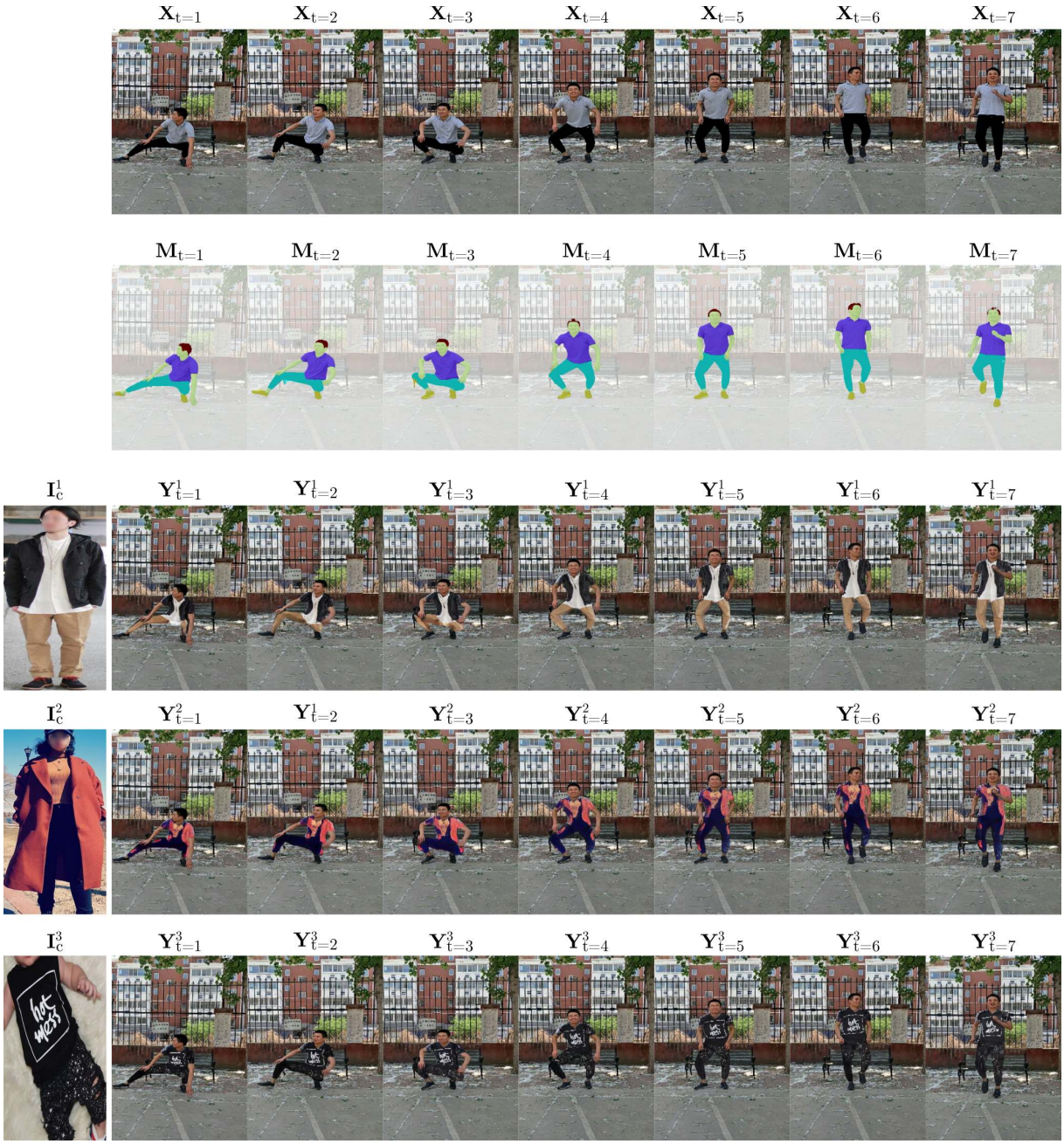


Figure 4.29: **Transferring clothes in real videos.** Sequence #1.

Figure 4.30: **Transferring clothes in real videos.** Sequence #2.

Figure 4.31: **Transferring clothes in real videos.** Sequence #3.

from the input one and require hallucinating large portions of the image.

For facial expression editing we have presented a novel GAN model for face animation in the wild that can be trained in a fully unsupervised manner. It advances current works which, so far, had only addressed the problem for discrete emotions category editing and portrait images. Our model encodes anatomically consistent face deformations parameterized by means of AUs. Conditioning the GAN model on these AUs allows the generator to render a wide range of expressions by simple interpolation. Additionally, we embed an attention model within the network which allows focusing only on those regions of the image relevant for every specific expression. By doing this, we can easily process images in the wild, with distracting backgrounds and illumination artifacts. We have exhaustively evaluated the model capabilities and limits in the EmotioNet and RaFD datasets as well as in images from movies. The results are very promising, and show smooth transitions between different expressions in the wild.

Finally, we have presented the first system capable of photo-realistically transferring clothes from a reference image to video sequences that can be trained in a non-supervised manner. Our architecture is based on a GAN equipped with a physical memory that stores and maintains an initially incomplete texture map that is progressively filled with the new inferred occluded parts. This memory is designed so that the model can retrieve the texture of prior frames while still being able to update it with new estimations. This strategy allows encoding long term dependencies in the texture map that can then be consistently transferred to long video sequences. The results are visually appealing and we believe they open the possibility to be used in the future as a quick virtual try-on system for clothing.

5

From 3D Reconstruction to Synthesis and Back

5.1 Introduction

Throughout this dissertation we have studied the problem of 3D data understanding from two different perspectives. First, in Chapter 3, we addressed the problem of explicit 3D reconstruction. Then, in Chapter 4, we presented generative methods able to implicitly model 3D data. This chapter combines the two previous chapters approaches into one, bridging the gap between reconstruction and synthesis by proposing two general-purpose models for both reconstructing and synthesising novel images.

We start introducing C-Flow (Sec. 5.3), a novel learning method capable of generalizing to large datasets based on flow-based generative models. Recall that in Section 2.3 we presented their main formulation and properties. As a brief summary, standard flow models have highly desirable properties like exact log-likelihood evaluation and exact latent-variable inference, but they are still in their infancy and have not received as much attention as alternative generative models. C-Flow is a novel conditioning scheme that brings normalizing flows to an entirely new scenario with great possibilities for multi-modal data modeling. C-Flow is based on a parallel sequence of invertible mappings in which a source flow guides the target flow at every step, enabling fine-grained control over the generation process. We also devise a new strategy to model unordered 3D point clouds that, in combination with the conditioning scheme, makes it possible to address 3D reconstruction from a single image and its inverse problem of rendering an image given a point cloud. We demonstrate our conditioning method to be very adaptable, being also applicable to image manipulation, style transfer and multi-modal image-to-image mapping in a diversity of domains, including RGB images, segmentation maps and edge masks.

Secondly, we propose a completely different approach. Instead of trying to generalize to large datasets as in C-Flow, we present D-NeRF (Sec. 5.4), a neural rendering radiance field technique specifically designed to model one scene at a time with very fine detail. Neural

radiance field techniques combine machine learning with classic graphic renders by training a deep network to map 5D input coordinates (representing spatial location and viewing direction) into a volume density and view-dependent emitted radiance. D-NeRF extends neural radiance fields to a dynamic domain, allowing to reconstruct and render novel images of objects under rigid and non-rigid motions from a *single* camera moving around the scene. For this purpose we consider time as an additional input to the system, and split the model two main sub-models: one that encodes the scene into a canonical space and another that maps this canonical representation into the deformed scene at a particular time. Both mappings are simultaneously learned using fully-connected networks. Once the networks are trained, D-NeRF can render novel images, controlling both the camera view and the time variable, and thus, the object movement. We demonstrate the effectiveness of D-NeRF on scenes with objects under rigid, articulated and non-rigid motions.

5.2 Related Work

We next elaborate on the state-of-the-art of the main building blocks for simultaneous reconstruction and synthesis: generative models and implicit data representation.

Generative Models for Simultaneous Reconstruction and Synthesis

As we have seen in previous chapters, the success of Deep Learning has spurred a large number of approaches for 3D reconstruction [23, 297–300]. All 3D reconstruction techniques presented till now on this disertation are discriminative, meaning that they learn direct mappings between output shapes and input images. Generative models, in contrast, capture the actual shape distribution from the training set, enabling not only to reconstruct new test images, but also to sample new shapes from the learned distribution. In the first part of this chapter we are interested in combining reconstruction and synthesis via generative models. There exist several works along this line, mainly based on GANs and VAEs. For instance, GANs have been used in Wu *et al.* [301] to model objects in a voxel representation; Hamu *et al.* [302] used them to model body parts; and Pumarola *et al.* [24] to learn the manifold of geometry images representing clothed 3D bodies. VAEs [303–306] have also been applied to model 3D data. Joon Park *et al.* [17] used auto-decoders [307, 308] to represent the surface of a shape with a continuous volumetric field.

While GANs and VAEs are the most studied deep generative models so far, we belief that flow-based generative models [41, 42] are key for 3D data modeling as they offer very attractive properties such as the ability to estimate exact log-likelihood, efficient synthesis and exact

latent-variable inference. Further advances have been proposed in RealNVP [309] by introducing the affine coupling layers and in Glow [12], through an architecture with 1x1 invertible convolutions for image generation and editing. These works have been later applied to audio generation [310–313], image modeling [314–316] and video prediction [317].

In our case it is crucial that we can condition the generation process (*e.g.*for 3D reconstruction we want to condition the 3D data generation on an input image of the desired scene to be reconstructed). Some recent works have proposed strategies for conditioning normalizing flows by combining them with other generative models. For instance, [315,318] combine flows with GANs. These models, however, are more difficult to train as adversarial losses tend to introduce instabilities. Similarly, for the specific application of video prediction, [317] enforces an autoregressive model onto the past latent variables to predict them in the future. Dual-Glow [314] uses a conditioning scheme for MRI-to-PET brain scan mapping by concatenating the prior distribution of the source image with the latent variables of the target image. In Sec. 5.3, we introduce a novel mechanism to condition flow-based generative models enabling fine-grained control for reconstruction and render tasks.

Implicit Models for Simultaneous Reconstruction and Synthesis

In the second part of this chapter we focus on another major trend for 3D modeling based on implicit functions. In Chapter 3 we have already seen [28,155], voxels [151,319], octrees [152, 153] and geometry images [23]. In this last chapter we exploit a very recent technique, implicit functions for representing 3D data in an implicit manner via a neural network [17,305, 320–323]. The main idea behind this approach is to describe the information (*e.g.*occupancy, distance to surface, color, illumination) of a 3D point $\mathbf{x}$ as the output of a neural network $f(\mathbf{x})$. Compared to the previously mentioned representations, neural implicit representations allow for continuous surface reconstruction at a low memory footprint.

The first works exploiting implicit representations [17,305,320,321] for 3D representation were limited by their requirement of having access to 3D ground-truth geometry, often expensive or even impossible to obtain for in the wild scenes. Subsequent works relaxed this requirement by introducing a differentiable render allowing 2D supervision. For instance, [324] proposed an efficient ray-based field probing algorithm for efficient image-to-field supervision. [325,326] introduced an implicit-based method to calculate the exact derivative of a 3D occupancy field surface intersection with a camera ray. In [327], a recurrent neural network was used to ray-cast the scene and estimate the surface geometry. NeRF [328] showed that by implicitly representing a rigid scene using 5D radiance fields makes it possible to capture high-resolution geometry and photo-realistically rendering novel views. [329] extended this method to handle variable illumination and transient occlusions to deal with in the wild images. In [330], even

more complex 3D surfaces were represented by using voxel-bouded implicit fields. And [331] circumvented the need of multiview camera calibration.

However, while all mentioned methods achieve impressive results on rigid scenes, none of them can deal with dynamic and deformable scenes. Occupancy flow [332] was the first work to tackle non-rigid geometry by learning continuous vector field assigning a motion vector to every point in space and time, but it requires full 3D ground-truth supervision. Neural volumes [333] produced high quality reconstruction results via an encoder-decoder voxel-based representation enhanced with an implicit voxel warp field, but they require a muti-view image capture setting. In Sec. 5.4 we present D-NeRF, is the first approach able to generate a neural implicit representation for non-rigid and time-varying scenes, trained solely on monocular data without the need of 3D ground-truth supervision nor a multi-view camera setting.

5.3 Generative Flow Model for Images and 3D Data

Generative models have become extremely popular in the machine learning and computer vision communities. Two main actors currently prevail in this scenario, VAEs [39] and especially GANs [40]. In this section we focus on a different family, the so-called flow-based generative models [41], which remain under the shadow of VAEs and GANs despite offering very appealing properties. Compared to other generative method, flow-based models build upon a sequence of reversible mappings between the input and latent space that allow for (i) exact latent-variable inference and log-likelihoood evaluation, (ii) efficient and parallelizable inference and synthesis and (iii) useful and simple data manipulation by operating directly on the latent space. A detailed explanation of flow-based generative models has been already given in the Overview chapter (Sec. 2.3).

The main contribution of this section is a novel approach to bridge the gap between reconstruction and synthesis by proposing a single general-purpose model capable of both reconstruction and synthesis. For this purpose, we introduce C-Flow, a novel conditional normalizing flows, making it possible to perform multi-modality transfer tasks which have so far not been explored under the umbrella of flow-based generative models. C-Flow consists on two parallel flow branches, interconnected across their reversible functions using *conditional coupling layers* and trained with an *invertible cycle consistency loss*. This scheme allows guiding a source domain towards a target domain guaranteeing the satisfaction of the aforementioned properties of flow-based models. Conditional inference is then implemented in a simple manner, by (exactly) embedding the source sample into its latent space, sampling a point from a Gaussian prior, and then propagating them through the learned normalizing flow. For example, for the application of synthesizing multiple plausible photos given a semantic segmentation mask, each image is

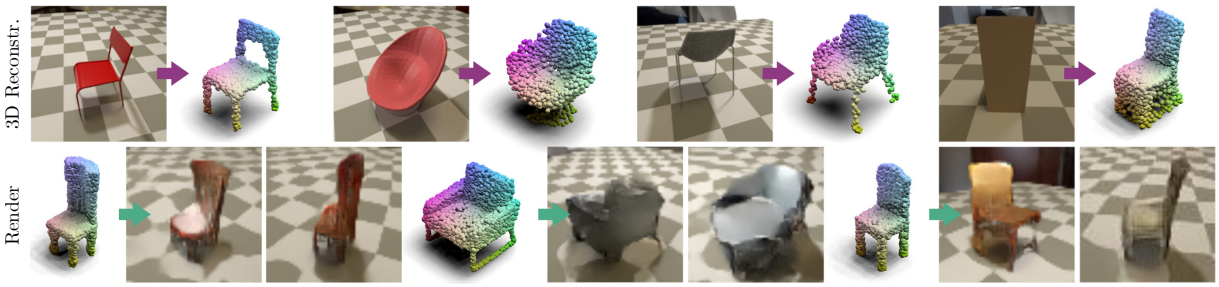


Figure 5.1: **C-Flow overview.** We propose C-Flow, a conditioning scheme for flow-based generative models applicable to many different domains. The figure shows the results of modeling the conditional distributions *image* $\leftrightarrow$ *3D point cloud*. In the top row we apply this model for 3D reconstruction (*image* $\rightarrow$ *point cloud*), and in the bottom row for rendering new images (*point cloud* $\rightarrow$ *image*). Our model allows sampling multiple times from this conditional distribution to generate several renderings of the same point cloud.

generated by jointly propagating the segmentation embedding and a random point drawn from a prior distribution across the learned flow.

Our second contribution is a strategy to enable flow-based methods to model unordered 3D point clouds. Specifically, we introduce (i) a re-ordering of the 3D data points according to a Hilbert sorting scheme, (ii) a global feature operation compatible with the reversible scheme, and (iii) an invertible cycle consistency that penalizes the Chamfer Distance (CD). Combining this strategy with the proposed conditional scheme we can then address tasks such as shape interpolation, 3D object reconstruction from an image, and rendering an image given a 3D point cloud (Fig. 5.1).

Importantly, our new conditioning scheme enables a wide range of tasks beyond 3D point cloud modeling. In particular, we are the first flow-based model to show mapping between a large diversity of domains, including image-to-image, pointcloud-to-image, edges-to-image segmentation-to-image and their inverse mappings. Also, we are the first to demonstrate application in image content manipulation and style transfer tasks. We believe our conditioning scheme, and its ability to deal with a variety of domains, opens the door to building general-purpose and easy to train solutions.

In summary, this section main contributions are: 1) a novel conditional normalizing flows formulation to bridge the gap between reconstruction and synthesis by proposing a single general-purpose model capable of both reconstruction and synthesis; and 2) a new strategy to model unordered 3D point clouds with normalizing flows.

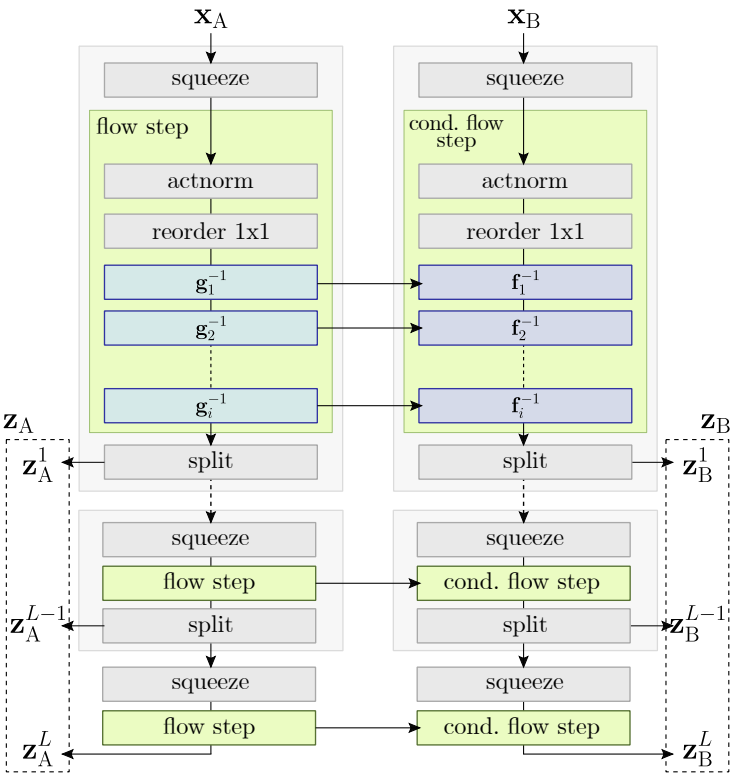


Figure 5.2: **The C-Flow model** consists of two parallel flow branches mutually interconnected with conditional coupling layers. This scheme allows sampling $\mathbf{x}_B$ conditioned on $\mathbf{x}_A$. For a detailed description on functions in grey refer to [12].

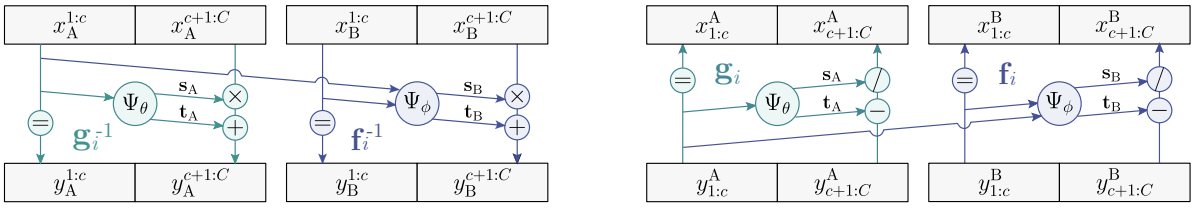
5.3.1 Conditional Flow-Based Generative Model

We next extend the original formulation of flow-based generative models (explained in Sec. 2.3) to model conditional distributions. Let us define a true data distribution $(\mathbf{x}_A, \mathbf{x}_B) \sim p^*(\mathbf{x}_A, \mathbf{x}_B)$. Our goal is to learn a model for $\mathbf{x}_B \sim p^*(\mathbf{x}_B|\mathbf{x}_A)$ to map sample points from domain A to domain B. For example, for the application of 3D reconstruction from a single view, $\mathbf{x}_A$ would be an image and $\mathbf{x}_B$ a 3D point cloud. To this end, we propose a conditional flow-based generative model extending the architectures of [12, 309]. Our L-levels model, learns both distributions with two bijective transformations $\mathbf{g}_\theta$ and $\mathbf{f}_\phi$ (Fig. 5.2):

$$\mathbf{z}_A \sim p_\vartheta(\mathbf{z}_A), \quad \mathbf{z}_B \sim p_\varphi(\mathbf{z}_B) \tag{5.1}$$

$$\mathbf{x}_A = \mathbf{g}_\theta(\mathbf{z}_A), \quad \mathbf{x}_B = \mathbf{f}_\phi(\mathbf{z}_B|\mathbf{z}_A) \tag{5.2}$$

$$\mathbf{z}_A = \mathbf{g}_\theta^{-1}(\mathbf{x}_A), \quad \mathbf{z}_B = \mathbf{f}_\phi^{-1}(\mathbf{x}_B|\mathbf{x}_A) \tag{5.3}$$



(a) Forward Propagation (b) Backward Propagation

Figure 5.3: **Conditional coupling layer for forward and backward propagation.** Given two input tensors $\mathbf{x}_A$ and $\mathbf{x}_B$, the proposed conditional coupling layer transforms the second half of $\mathbf{x}_B$ conditioned on the first halves of $\mathbf{x}_A$ and $\mathbf{x}_B$. The first halves of all tensors are not updated. By sequentially concatenating these bijective operations we can transform data points x into their latent representation y (forward propagation) and vice versa (backward propagation).

where $\mathbf{z}_A$ and $\mathbf{z}_B$ are latent-variables, and $p_{\boldsymbol{\vartheta}}(\mathbf{z}_A)$ and $p_{\boldsymbol{\varphi}}(\mathbf{z}_B)$ are tractable spherical multivariate Gaussian distributions with learnable mean and variance.

We then define the mapping $\mathcal{M}$ to sample $\mathbf{x}_B$ conditioned on $\mathbf{x}_A$, as a three-step operation:

$$\mathbf{z}_A = \mathbf{g}_{\boldsymbol{\theta}}^{-1}(\mathbf{x}_A) \qquad \text{encode condition } \mathbf{x}_A \tag{5.4}$$

$$\mathbf{z}_B \sim p_{\boldsymbol{\varphi}}(\mathbf{z}_B) \qquad \text{sample latent-variable } \mathbf{z}_B \tag{5.5}$$

$$\mathbf{x}_B = \mathbf{f}_{\boldsymbol{\phi}}(\mathbf{z}_B|\mathbf{z}_A) \qquad \text{generate } \mathbf{x}_B \text{ conditioned on } \mathbf{x}_A \tag{5.6}$$

In the following we describe how this conditional framework is implemented.

Conditional Coupling Layer. When designing the conditional coupling layer we need to fulfill the constraint that each transformation has to be bijective and tractable. As shown in [41, 309], both these issues can be overcome by choosing transformations with triangular Jacobian. In this case their determinant is calculated as the product of diagonal terms, making the computation tractable and ensuring invertibility. Motivated by these works, we propose an extension of their coupling layer to account for cross-domain conditioning. A schematic of the proposed layers is shown in Fig. 5.3. Formally, let us define $y \triangleq h_i$ and $x \triangleq h_{i-1}$. We then write the invertible function $\mathbf{f}^{-1}$ to transform a data point $\mathbf{x}_B$ based on $\mathbf{x}_A$ as follows:

$$\begin{cases} y_B^{1:c} &= x_B^{1:c} \\ y_B^{c+1:C} &= x_B^{c+1:C} \odot \exp\left(s\left(x_A^{1:c}, x_B^{1:c}\right)\right) + t\left(x_A^{1:c}, x_B^{1:c}\right) \end{cases} ,$$

where C is the number of channel dimensions in both data points, $\odot$ denotes element-wise multiplication and s and t are the scale and translation functions from $(\mathbb{R}^c, \mathbb{R}^c) \mapsto \mathbb{R}^{C-c}$. We set $c = C/2$ in all experiments.

The inverse $\mathbf{f}$ of the conditional coupling layer is:

$$\begin{cases} x_{\mathrm{B}}^{1:c} &= y_{\mathrm{B}}^{1:c} \\ x_{\mathrm{B}}^{c+1:C} &= \left(y_{\mathrm{B}}^{c+1:C} - t\left(y_{\mathrm{A}}^{1:c}, y_{\mathrm{B}}^{1:c}\right)\right) \odot \exp\left(s\left(x_{\mathrm{A}}^{1:c}, x_{\mathrm{B}}^{1:c}\right)\right), \end{cases} \tag{5.7}$$

and its Jacobian:

$$\frac{\partial y_{\mathrm{B}}}{\partial x^{\top}} = \begin{bmatrix} \mathbf{I}_c & \mathbf{0} \\ \frac{\partial y_{\mathrm{B}}^{c+1:C}}{\partial (x^{1:c})^{\top}} & \mathrm{diag}\left(\exp\left(s\left(x_{\mathrm{A}}^{1:c}, x_{\mathrm{B}}^{1:c}\right)\right)\right) \end{bmatrix}, \tag{5.8}$$

where $\boldsymbol{I}_c \in \mathbb{R}^{c \times c}$ is an identy matrix. Since the Jacobian is a triangular matrix, its determinant can be calculated efficiently as the product of the diagonal elements. Note that it is not required to compute the Jacobian of the functions s and t, enabling them to be arbitrarily complex. In practice, we implement these functions using a Convolutional Neural Network $\Psi(\cdot)$ that returns both $\log(\mathbf{s})$ and $\mathbf{t}$.

Coupling Network Architecture. We next describe the architecture of $\Psi_{\boldsymbol{\theta}}(\cdot)$ and $\Psi_{\boldsymbol{\phi}}(\cdot)$ used to regress the affine transform applied at every conditional coupling layer at each $\mathbf{g}_i$ and $\mathbf{f}_i$ respectively. We build upon the stack of three 2D convolution layers proposed by [12]. The first two layers have a filter size of 3×3 and 1×1 with 512 output channels followed by actnorm [12] and a ReLU activation. The third layer regresses the final scale and translation by applying a 2D convolutional layer with filter size 3×3 initialized with zeros such that each affine transformation at the beginning of training is equivalent to an identity function.

For the transformation $\mathbf{g}_i^{-1}(x_{\mathrm{A}})$ we exactly use this architecture, but for $\mathbf{f}_i^{-1}(x_{\mathrm{B}}|x_{\mathrm{A}})$ we extend it to take into account the conditioning x_{A}. Concretely, in $\mathbf{f}_i^{-1}$, x_{B} is initially transformed by two convolution layers, like the first two of $\mathbf{g}_i^{-1}$. Then, x_{A} is adapted with a channel-wise affine transform implemented by a 1×1 convolution. Finally, its output is added to the transformed x_{B}. To ensure a similar contribution of x_{A} and x_{B} their activations are normalized with actnorm so that they operate in the same range. A final 3×3 convolution regresses the conditional coupling layer operators $\log(\mathbf{s}_{\mathrm{B}})$ and $\mathbf{t}_{\mathrm{B}}$.

5.3.2 Learning the Model

Invertible Cycle Consistency. We train our model to maximize the log-likelihood of the training dataset. However, likewise in GANs learning [51, 52], we found beneficial to add a loss encouraging the generated and real samples to be similar in L1. To do so, we exploit the fact that our model is made of bijective transformations, and introduce what we call an invertible

cycle consistency. This operation can be summarized as follows:

$$\{\mathbf{x}_\mathrm{A}, \mathbf{x}_\mathrm{B}\} \xrightarrow{\mathbf{g}^{-1},\mathbf{f}^{-1}} \{\mathbf{z}_\mathrm{A}, \mathbf{z}_\mathrm{B}\} \rightarrow \{\mathbf{z}_\mathrm{A}, \hat{\mathbf{z}}_\mathrm{B}\} \xrightarrow{\mathbf{f}} \hat{\mathbf{x}}_\mathrm{B}. \tag{5.9}$$

Concretely, the data points observations ($\mathbf{x}_\mathrm{A}$, $\mathbf{x}_\mathrm{B}$) are initially mapped into their latent variables ($\mathbf{z}_\mathrm{A}$, $\mathbf{z}_\mathrm{B}$), where each variable is composed of an L-level stack. As demonstrated in [309] the first levels encode the high frequencies (details) in the data, and the last levels the low frequencies.

We then resample the first $L-1$ dimensions of $\mathbf{z}_\mathrm{B}$ from a Gaussian distribution, *i.e.* $\mathbf{z}_\mathrm{B} = [\mathbf{z}_1, \ldots, \mathbf{z}_L] \rightarrow \hat{\mathbf{z}}_\mathrm{B} = [\mathcal{N}(0, \boldsymbol{I})_1, \ldots, \mathcal{N}(0, \boldsymbol{I})_{L-1}, \mathbf{z}_L]$. By doing this, $\hat{\mathbf{z}}_\mathrm{B}$ is only retaining the lowest frequencies of the original $\mathbf{z}_\mathrm{B}$.

As a final step, we invert $\mathbf{f}^{-1}$, to recover $\hat{\mathbf{x}}_\mathrm{B} = \mathbf{f}(\hat{\mathbf{z}}_\mathrm{B}|\mathbf{z}_\mathrm{A})$ and penalize its L1 difference w.r.t the original $\mathbf{x}_\mathrm{B}$. What we are essentially doing is to force the model to use information from the condition $\mathbf{x}_\mathrm{A}$ so that the recover sample $\hat{\mathbf{x}}_\mathrm{B}$ is as similar as possible to the original $\mathbf{x}_\mathrm{B}$. Note that if reconstructed $\hat{\mathbf{z}}_\mathrm{B}$ based on the entire latent variable, the recovered sample would be identical to the original $\mathbf{x}_\mathrm{B}$ because $\mathbf{f}$ is bijective, and this loss would be meaningless.

Total Loss. Formally, denoting the training pairs of observations as $\{\mathbf{x}_\mathrm{A}^{(i)}, \mathbf{x}_\mathrm{B}^{(i)}\}_{i=1}^N$, the model parameters are learned by minimizing the following loss function:

$$\frac{1}{N}\sum_{i=1}^{N}\left[-\log p_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}_\mathrm{A}^{(i)}, \mathbf{x}_\mathrm{B}^{(i)}) + \lambda \left\|\mathbf{x}_\mathrm{B}^{(i)} - \hat{\mathbf{x}}_\mathrm{B}^{(i)}\right\|_1\right] \tag{5.10}$$

The first term maximizes the joint likelihood of the data observations. With our design, it also maximizes the conditional likelihood of $\mathbf{x}_\mathrm{B}|\mathbf{x}_\mathrm{A}$ and thus forces the model to learn the desired mapping. To show this, we apply the law of total probability and we factor it into:

$$-\sum_{i=1}^{N}\log p_{\boldsymbol{\theta}}(\mathbf{x}_\mathrm{A}^{(i)}) - \sum_{i=1}^{N}\log p_{\boldsymbol{\phi}}(\mathbf{x}_\mathrm{B}^{(i)}|\mathbf{x}_\mathrm{A}^{(i)}) \tag{5.11}$$

Due to the diagonal structure of the Jacobians, the marginal likelihood of $\mathbf{x}_\mathrm{A}$ depends only on $\boldsymbol{\theta}$ (first sum), while the conditional of $\mathbf{x}_\mathrm{B}|\mathbf{x}_\mathrm{A}$ – only on $\boldsymbol{\phi}$. Maximizing the joint likelihood thus maximizes both likelihoods independently. The second term in Eq. (5.10) minimizes the cycle consistency loss. λ is a hyper-parameter balancing the terms. This loss is fully differentiable, and we provide details on how we optimize it in Sec. 5.3.3.

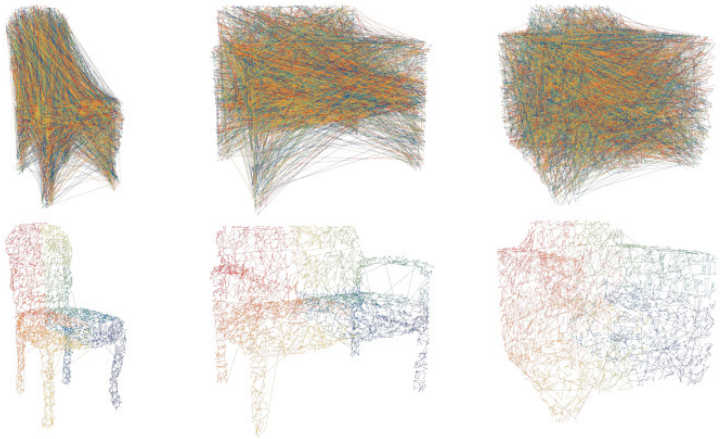


Figure 5.4: **Sorting 3D point clouds.** Point clouds corresponding to three different chairs. The colored line connects all points based on their ordering. **Top:** Unordered. **Bottom:** Applying the proposed sorting strategy. Note how the coloring is consistent across samples even for point clouds with different topology.

5.3.3 Modeling Unordered 3D Point Clouds

The model described so far can handle input data represented on regular grids but it fails to model unordered 3D point clouds, whose lack of spatial neighborhood ordering prevents convolutions from being applied. To process point clouds with deep networks, a common practice is to apply *symmetry operations* [334] that create fixed-size tensors of global features describing the entire point cloud sample. These operations require extracting point-independent features followed by a max-pool, which is not invertible and not applicable to normalizing flows. Another alternative would be the graph convolutional networks [335], although their high computational cost makes them not suitable for our scheme of multiple coupling layers. We propose a three-step mechanism to enable modeling 3D point clouds:

(i) Approximate Sorting with Space-Filling Curves. C-Flow is based on convolutional layers which require input data with a local neigboorhood consistent across samples. To fulfill this condition on unordered point clouds, we propose to sort them based on proximity. As discussed in [334], for high dimensional spaces it is not possible to produce a perfect ordering stable to point perturbations. We therefore consider using the approximation provided by the Hilbert's space-filling curve algorithm [336]. For each training sample, we project its points into a 3D Hilbert curve and reorder them based on their ordering along the curve (Fig. 5.4). Notice that not only we can establish a neighborhood relationship but also a semantically-stable ordering (*e.g.*in Fig. 5.4 the chair's right-leg is always blue). To the best of our knowledge there is no previous work using such preprocessing for point clouds.

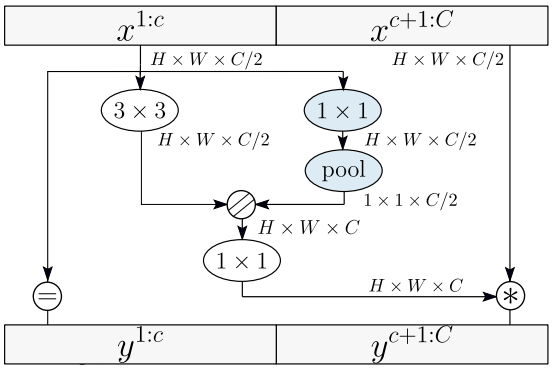


Figure 5.5: **Approximating global features in point clouds.** When dealing with point clouds (reordered and reshaped to a $H \times W \times 3$ size and using $c = C/2$) we approximate, with operations in blue, global features in coupling layers while still being invertible. $\circledast$ stands for affine transformation where the first $C/2$ input channels are the scale and the other half the translation.

(ii) Approximating Global Features. Hilbert Sort is not sufficient to model 3D data because of a major issue: it splits the space into equally sized quadrants and the Hilbert curve will cover all points in a quadrant before moving to the next. As a consequence, two points that were originally close in space, but lie near the boundaries of two different quadrants, will end up far away in the final ordering. To mitigate this effect we extend the proposed coupling network architecture (Sec. 5.3.1) with an approximate but invertible version of the global features proposed in [334] that describe the whole point cloud. Concretely, we first resample and reshape the reordered point cloud to form $H \times W \times 3$ matrices (in practice we use the same size as that of the images). Then we approximate the global descriptors of [334] through a 1×1 convolution to extract point-independent features followed by a max-pool applied only over the first half of the point cloud features $x^{1:c}$ (Fig. 5.5). The coupling layer remains bijective because during the backward propagation the approximated global features can be recovered using a similar strategy as in Eq. (5.7).

(iii) Symmetric Chamfer Distance for Cycle Consistency. For the specific case of point clouds, we observed that when penalizing the invertible cycle consistency with L1 the model converged to a mean Hilbert curve. Therefore, for point clouds, we substitute L1 by the symmetric Chamfer Distance (CD), which computes the mean Euclidean distance between the ground-truth point cloud $\mathbf{x}_{\mathrm{B}}$ and the recovered $\hat{\mathbf{x}}_{\mathrm{B}}$.

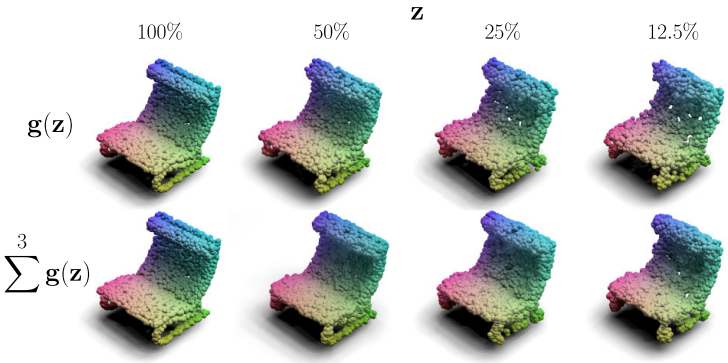


Figure 5.6: **Embedding 3D points clouds. Top:** Reconstruction with partial embeddings. **Bottom:** Reconstruction with three iterations of backward propagations of partial embeddings.

Implementation Details

Due to memory restrictions, we train with image samples of 64×64 resolution. For 3D point clouds, to maintain the same architecture as in images, we reshape each point cloud sample (list of 64^2 points) to 64×64. At test time we also regress 64^2 3D points per forward pass. Our implementation builds upon that of Glow [12]. We use Adam with learning rate $1e^{-6}$, $\beta_1 = 0.85$, $\beta_2 = 0.007$ and batch size 4. The multi-scale architecture consists of $L = 4$ levels with 12 flow steps per level ($K = 4 * 12$ in Eq. (2.6)) each and $2\times$ squeezing operations. For conditional sampling we found additive coupling ($s(\cdot) = 1$) to be more stable during training than affine transformation. The prior distributions $p_{\vartheta}(\mathbf{z}_{\mathrm{A}})$ and $p_{\varphi}(\mathbf{z}_{\mathrm{B}})$ are initialized with mean 0 and variance 1. The rest of weights are randomly initilized from a normal distribution with mean 0 and std 0.05. $\lambda = 10$ in Eq. (5.10). As in previous likelihood-based generative models [12,337], we observed that sampling from a reduced-temperature prior improves the results. To do so, we multiply the variance of $p_{\varphi}(\mathbf{z}_{\mathrm{B}})$ by $T = 0.9$. The model is trained with 4 GPUs P-100 for 10 days.

5.3.4 Experimental Evaluation

We next evaluate our system on diverse tasks: (i) Modeling point clouds, (ii) 3D reconstruction and rendering, (iii) Image-to-image mapping in a variety of domains and datasets, and (iv) Image manipulation and style transfer.

Modeling 3D Point Clouds

We evaluate the potential of our approach to model 3D point clouds on ShapeNet [338]. For this task, we do not consider the full conditioning scheme and only use one of the branches of C-Flow in Fig. 5.2, which we denote as C-Flow*.

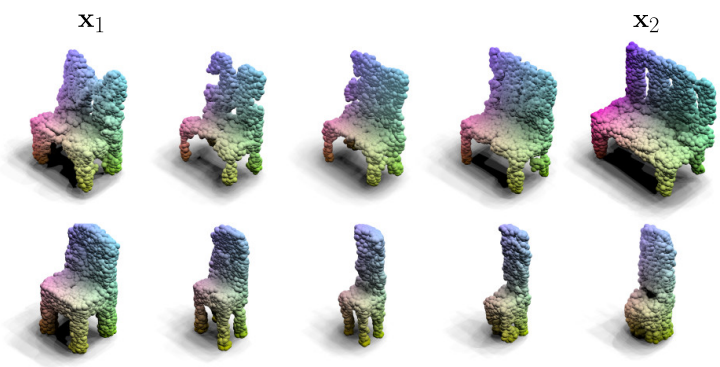


Figure 5.7: **Interpolation.** Results of interpolating two 3D point clouds $\mathbf{x}_1$ and $\mathbf{x}_2$ in the learned latent space.

Method	100%	50%	25%	12.5%
C-Flow* ≡ Glow [12]	0.00	0.39	0.39	0.39
C-Flow* + Sort	0.00	0.19	0.21	**0.22**
C-Flow* + Sort + GF-Coupling	**0.00**	**0.14**	**0.18**	0.31
AtlasNet-Sph. [300]		0.75		
AtlasNet-25 [300]		0.37		
DeepSDF [17]		0.20		

Table 5.1: **Representing 3D point clouds**. Chamfer Distance when recovering point clouds with partial embeddings. For all C-Flow* we change the embedding size at test, with no further training. The percentages are with respect to the input dimension (4096). For AtlasNet and DeepSDF we provide the results from [17].

In our first experiment we study the representation capacity of unknown shapes, formally defined as the ability to retain the information after mapping forward and backward between the original and latent spaces. For this purpose, we first map a real point cloud $\mathbf{x}$ to the latent space $\mathbf{z} = \mathbf{g}_{\boldsymbol{\theta}}^{-1}(\mathbf{x})$. The full-size embedding $\mathbf{z} = [\mathbf{z}_1, \ldots, \mathbf{z}_L]$ has as many dimensions as the input (HWC). Then we progressively remove information from $\mathbf{z}$ by replacing their left-most l components with samples drawn from a Gaussian, *i.e.* $\hat{\mathbf{z}} = [\mathcal{N}(0, \mathbf{I})_1, \ldots, \mathcal{N}(0, \mathbf{I})_l, \mathbf{z}_{l+1}, \ldots, \mathbf{z}_L]$. Note that the embedding size $L - l$ can be set at test time with no need to retrain, making tasks like point cloud compression straightforward. Finally we map back this embedding to the original point cloud space $\hat{\mathbf{x}} = \mathbf{g}_{\boldsymbol{\theta}}(\mathbf{z})$ and compare to $\mathbf{x}$.

Table 5.1 reports the Chamfer Distance for different embedding sizes. The plain version of C-Flow* (no conditioning, no sorting, no global features) is equivalent to Glow [12]. This version is consistently improved when introducing the sorting and global features strategies (Sec. 5.3.3). The error decreases gracefully as we increase the embedding size, and importantly, when using the full size embedding we obtain a perfect recovering (Fig. 5.6-top). This is a

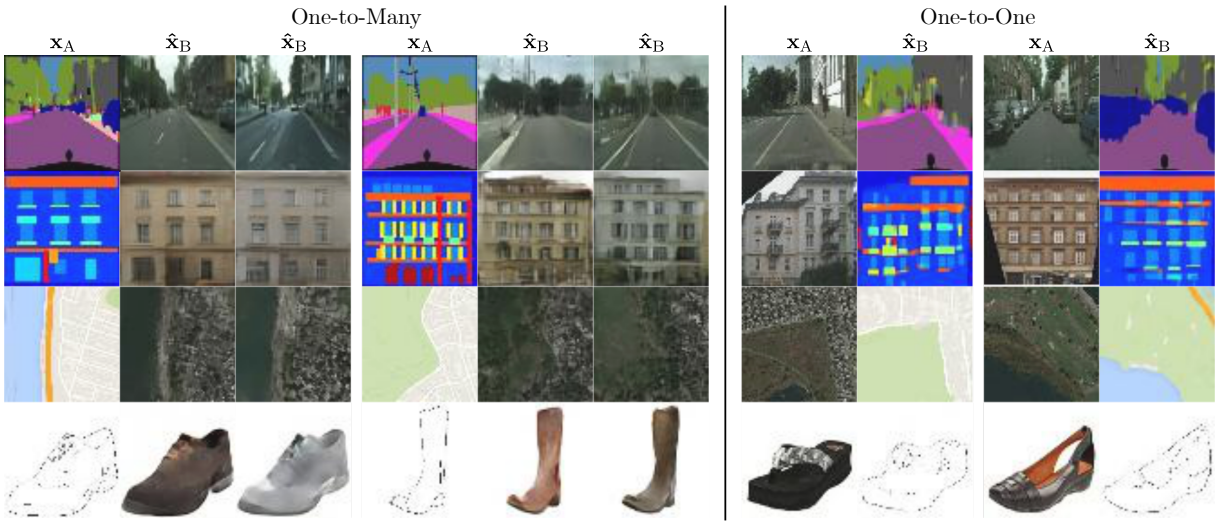


Figure 5.8: **Image-to-Image.** Results from 64×64 image-to-image mappings on a variety of domains. $\mathbf{x}_A$: source image; $\hat{\mathbf{x}}_B$: generated image in the target domain. The examples on the left correspond to target domains with high variability that when sampled multiple times generate different images. In the examples on the right the target domain has a small variability and the sampling becomes deterministic.

virtue of the bijective models, and is not a trivial property. Table 5.1 also reports the numbers of AtlasNet [300] and DeepSDF [17], showing that our approach achieves competitive results. This comparison is only indicative as the representation used in these approaches is inherently different ([300] parametric and [17] a continuous surface).

Recall that the left-most components randomly sampled in $\mathbf{z}$ encode the high details of the shape. We exploit this property to generate point clouds with an arbitrarily large number of points by performing multiple backward propagations ($\hat{\mathbf{x}} = \mathbf{g}_\theta(\hat{\mathbf{z}})$) of a partial embedding $\hat{\mathbf{z}}$ (Fig. 5.6-bottom). Every time we propagate, we recover a new set of 3D points allowing to progressively improve the density of the reconstruction.

Another task that can be addressed with C-Flow is shape interpolation in the latent space (Fig. 5.7).

3D Reconstruction & rendering

We next evaluate the ability of C-Flow to model the conditional distributions (i) *image* $\rightarrow$ *point cloud*, which enables to perform 3D reconstruction from a single image; and (ii) *point cloud* $\rightarrow$ *image*, which is its inverse problem of rendering an image given a 3D point cloud. Fig. 5.1 shows qualitative results on the *Chair* class of ShapeNet. In the top row our model is able to generate plausible 3D reconstructions of unknown objects even under strong self-occlusions (top-right

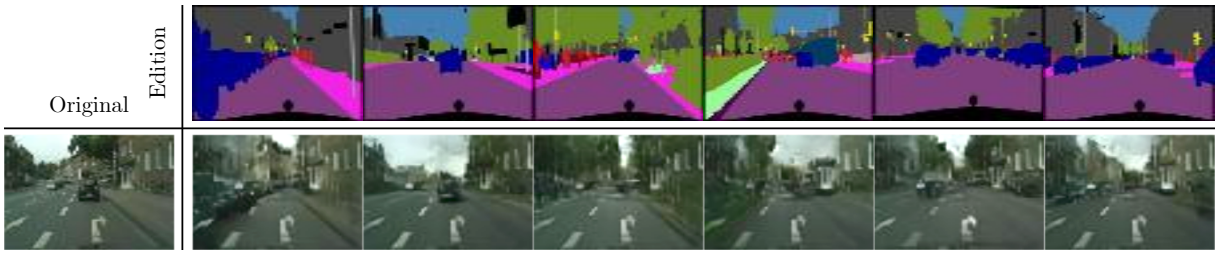
Method	Image → PC CD↓	Image ← PC BPD↓	IS↑
3D-R2N2 [297]	0.27	-	-
PSGN [303]	0.26	-	-
Pix2Mesh [298]	0.27	-	-
AtlasNet [300]	**0.21**	-	-
ONet [305]	0.23	-	-
C-Flow	0.86	4.38	1.80
C-Flow + Sort	0.52	**2.77**	2.41
C-Flow + Sort + GF-Coupling	0.49	2.87	**2.61**
C-Flow + Sort + GF-Coupling + CD	**0.26**	-	-

Table 5.2: **3D Reconstruction and rendering.** ↓: the lower the better, ↑: the higher the better. C-Flow is the first approach able to render images from point clouds. The same model can be used to perform 3D reconstruction from images. The results of all other methods are obtained from their original papers.

example). The second row depicts results for rendering, which highlights another advantage of our model: it allows sampling multiple times from the conditional distribution to produce several images of the same object which exhibit different properties (*e.g.*viewpoint or texture).

In Table 5.2 we compare C-Flow with other single-image 3D reconstruction methods 3D-R2N2 [297], PSGN [303], Pix2Mesh [298], AtlasNet [300] and ONet [305]. We evaluate 3D reconstruction in terms of the CD with the ground truth shapes. Our approach (last row) performs on par with [297, 298, 303] and it is slightly below the state-of-the-art techniques specifically designed for 3D reconstruction [300, 305].

With the same model, we can also render images from point clouds. To the best of our knowledge, no previous work can perform such mapping. While a few approaches do render point clouds [339–341], they hold on strong assumptions of knowing the RGB color per point and the camera calibration to project the point cloud onto the image plane. Table 5.2 also reports an ablation study about the different operations we devised to handle 3D point clouds, namely sorting the point cloud (Sort), approximating global features (GF-Coupling) and inverse cycle consistency with CD. In this case, evaluation is reported using IS [16] and Bits Per Dimension (BPD) which is equivalent to the negative log2-likelihood typically used to report flow-based methods performance. Results show a performance boost when using each of these components, and especially when combining them.



(a) Image content manipulation

(b) Style transfer

Figure 5.9: **Other applications.** Sample results on 64×64 image manipulation and style transfer. The model was not retrained for these tasks, and we used the same training weights to perform image-to-image in Fig. 5.8.

	C-Flow			C-Flow + cycle		
Method	BPD↓	SSIM↑	IS↑	BPD↓	SSIM↑	IS↑
segmentation → street views	3.21	0.37	1.80	**3.17**	**0.42**	**1.94**
segmentation ← street views	3.25	0.33	2.19	**3.05**	**0.36**	**2.23**
structure → facades	3.55	0.24	**1.92**	**3.54**	**0.26**	1.69
structure ← facades	3.55	**0.31**	**2.05**	3.55	0.30	2.01
map → aerial photo	3.65	**0.19**	1.52	3.65	0.17	**1.62**
map ← aerial photo	3.65	0.54	1.95	3.65	**0.57**	**1.97**
edges → shoes	1.70	0.66	2.40	**1.68**	**0.67**	**2.43**
edges ← shoes	1.65	0.64	1.61	1.65	**0.65**	**1.69**

Table 5.3: **Conditional image-to-image generation.** Evaluation of C-Flow (plain) and C-Flow + cycle consistency loss in image-to-image mapping.

Image-to-Image mappings

We evaluate the ability of C-Flow to perform multi-domain image-to-image mapping: *segmentation* ↔ *street views* trained on Cityscapes [342], *structure* ↔ *facade* trained on CMP Facades [343], *map* ↔ *aerial photo* trained on [52] and *edges* ↔ *shoes* trained on [52, 344, 345]. The examples on Fig. 5.8-left show mappings in which the target domain has a wide variance and multiple sampling generates different results (*e.g.*a semantic segmentation map can map to several grayscale images). The examples on the right have a target domain with a narrower variance, and despite multiple samplings the generated images are very similar (*e.g.*given an

image its segmentation is well defined).

Table 5.3 reports quantitative evaluations using SSIM [272], and again BPD and IS. When introducing the invertible cycle consistency loss (Sec. 5.3.2) the model does not improve its compression abilities (BPD) but improves in terms of structural similarity (SSIM) and semantic content (IS). It is worth to mention that while GANs have shown impressive image-to-image mapping results, even at high resolution [346], ours is the first work that can address such tasks using normalizing flows.

Other Applications

Finally, we demonstrate the versatility of C-Flow being the first flow-based method capable of performing style transfer and image content manipulation (Fig. 5.9). Importantly, the model was *not retrained* for these specific tasks, and we use the same parameters learned to perform image-to-image mappings (Sec. 5.3.4). For image manipulation we use the weights of *segmentation* $\rightarrow$ *street view* and for style transfer those of *edges* $\leftrightarrow$ *shoes*. Formally, let the domain A to be the structure (*e.g.*segmentation mask) and the domain B to be the image (*e.g.*street view). Then, image manipulation is achieved via three operations:

$$\mathbf{z}_{\mathrm{B}}^{1} = \mathbf{f}_{\phi}^{-1}(\mathbf{x}_{\mathrm{B}}^{1}|\mathbf{x}_{\mathrm{A}}^{1}) \qquad \text{encode original image } \mathbf{x}_{\mathrm{B}}^{1} \tag{5.12}$$

$$\mathbf{z}_{\mathrm{A}}^{2} = \mathbf{g}_{\boldsymbol{\theta}}^{-1}(\mathbf{x}_{\mathrm{A}}^{2}) \qquad \text{encode desired structure } \mathbf{x}_{\mathrm{A}}^{2} \tag{5.13}$$

$$\mathbf{x}_{\mathrm{B}}^{2} = \mathbf{f}_{\phi}(\mathbf{z}_{\mathrm{B}}^{1}|\mathbf{z}_{\mathrm{A}}^{2}) \qquad \text{synthesise new image } \mathbf{x}_{\mathrm{B}}^{2} \tag{5.14}$$

Note that following this generation approach we are no longer conditioning based only on A, and now the synthesised image is jointly conditioned on A (for structure) and B (for texture).

To perform style transfer, we first transform the content image into its structure $\mathbf{x}_{\mathrm{A}}^{2}$. For instance, in Fig. 5.9-bottom, the content of the *shoe* is initially mapped onto its *edge* structure with the *shoes* $\rightarrow$ *edges* weights. Then, we apply the same procedure as we did for image manipulation using the *edges* $\rightarrow$ *shoes* weights, setting $\mathbf{x}_{\mathrm{A}}^{1}$ to be the structure of the content image and $\mathbf{x}_{\mathrm{B}}^{1}$ the style image.

5.4 Neural Radiance Fields for Dynamic Scenes

In the previous section we have studied how to model a large 3D data corpus. In this section we no longer care about learning a manifold of plausible 3D objects but instead on how to model one specific scene at a time with extreme photo-realistic details. Photo-realistically modeling a scene from a sparse set of input images is necessary for many applications in *e.g.*augmented

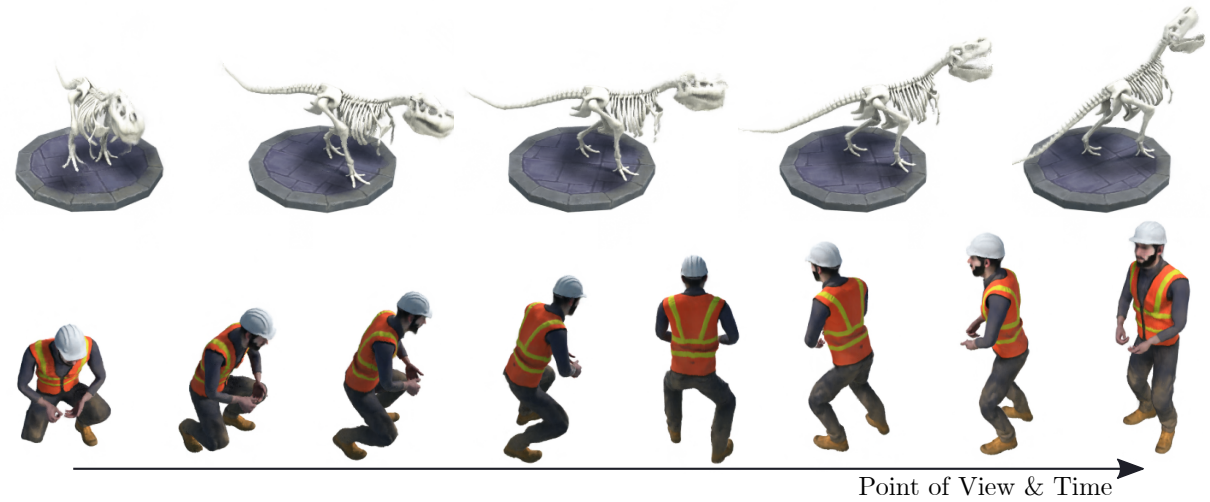


Figure 5.10: **D-NeRF overview.** We propose D-NeRF, a method for synthesizing novel views, at an arbitrary point in time, of dynamic scenes with complex non-rigid geometries. We optimize an underlying deformable volumetric function from a sparse set of input monocular views without the need of ground-truth geometry nor multi-view images. The figure shows two scenes under variable points of view and time instances synthesised by the proposed model.

reality, virtual reality, 3D content production, games and the movie industry. Recent advances in the emerging field of neural rendering, which learn scene representations encoding both geometry and appearance [325, 328–331, 347], have achieved results that largely surpass those of traditional Structure-from-Motion [348–350], light-field photography [351] and image-based rendering approaches [352]. For instance, the Neural Radiance Fields (NeRF) [328] have shown that simple multilayer perceptron networks can encode the mapping from 5D inputs (representing spatial locations (x, y, z) and camera views (θ, ϕ)) to emitted radiance values and volume density. This learned mapping allows then free-viewpoint rendering with extraordinary realism. Subsequent works have extended Neural Radiance Fields to images in the wild undergoing severe lighting changes [329] and have proposed sparse voxel fields for rapid inference [330]. Similar schemes have also been recently used for multi-view surface reconstruction [331] and learning surface light fields [353].

Nevertheless, all these approaches assume a *static* scene without moving objects. In this section we relax this assumption and propose, to the best of our knowledge, the first end-to-end neural rendering system that is applicable to dynamic scenes, made of both still and moving/deforming objects. While there exist approaches for 4D view synthesis [354], our approach is different in that: 1) we only require a single camera; 2) we do not need to pre-compute a 3D reconstruction; and 3) our approach can be trained end-to-end.

Our idea is to represent the input of our system with a continuous 6D function, which besides

3D location and camera view, it also considers the time component t. Naively extending NeRF to learn a mapping from (x, y, z, t) to density and radiance does not produce satisfying results, as the temporal redundancy in the scene is not effectively exploited. Our observation is that objects can move and deform, but typically do not appear or disappear. Inspired by classical 3D scene flow [355], the core idea to build our method, denoted Dynamic-NeRF (D-NeRF in short), is to decompose learning in two modules. The first one learns a spatial mapping $(x, y, z, t) \rightarrow (\Delta x, \Delta y, \Delta z)$ between each point of the scene at time t and a *canonical scene* configuration. The second module regresses the scene radiance emitted in each direction and volume density given the tuple $(x + \Delta x, y + \Delta y, z + \Delta z, \theta, \phi)$. Both mappings are learned with deep fully connected networks without convolutional layers. The learned model then allows to synthesize novel images, providing control in the continuum (θ, ϕ, t) of the camera views and time component, or equivalently, the dynamic state of the scene (see Fig. 5.10).

We thoroughly evaluate D-NeRF on scenes undergoing very different types of deformation, from articulated motion to humans performing complex body poses. We show that by decomposing learning into a canonical scene and scene flow D-NeRF is able to render high-quality images while controlling both camera view and time components. As a side-product, our method is also able to produce complete 3D meshes that capture the time-varying geometry and which remarkably are obtained by observing the scene under a specific deformation only from one single viewpoint.

In summary, this section main contributions are: 1) the first neural radiance field approach for modeling dynamic scenes that can be trained end-to-end from only a sparse set of images acquired with a moving camera, and 2) a new benchmark to evaluate techniques to model dynamic objects under large deformations and realistic non-Lambertian materials.

5.4.1 Problem Formulation

Given a sparse set of images of a dynamic scene captured with a monocular camera, we aim to design a deep learning model able to implicitly encode the scene and synthesize novel views at an arbitrary time (see Fig. 5.11).

Formally, our goal is to learn a mapping $\mathcal{M}$ that, given a 3D point $\mathbf{x} = (x, y, z)$, outputs its emitted color $\mathbf{c} = (r, g, b)$ and volume density σ conditioned on a time instant t and view direction $\mathbf{d} = (\theta, \phi)$. That is, we seek to estimate the mapping $\mathcal{M} : (\mathbf{x}, \mathbf{d}, t) \rightarrow (\mathbf{c}, \sigma)$.

An intuitive solution would be to directly learn the transformation $\mathcal{M}$ from the 6D space $(\mathbf{x}, \mathbf{d}, t)$ to the 4D space $(\mathbf{c}, \sigma)$. However, as we will show in the results subsection, we obtain consistently better results by splitting the mapping $\mathcal{M}$ into Ψ_x and Ψ_t, where Ψ_x represents the

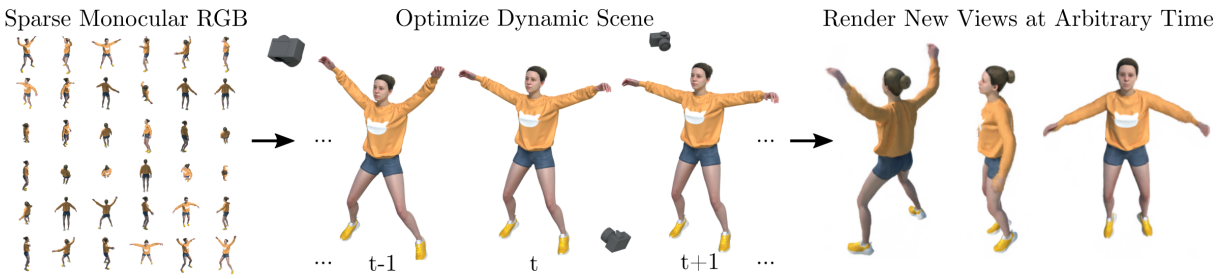


Figure 5.11: **D-NeRF problem definition.** Given a sparse set of images of a dynamic scene moving non-rigidly and being captured by a monocular camera, we aim to design a deep learning model to implicitly encode the scene and synthesize novel views at an arbitrary time. Here, we visualize a subset of the input training frames paired with accompanying camera parameters, and we show three novel views at three different time instances rendered by the proposed method.

scene in canonical configuration and Ψ_t a mapping between the scene at time instant t and the canonical one. More precisely, given a point $\mathbf{x}$ and viewing direction $\mathbf{d}$ at time instant t we first transform the point position to its canonical configuration as $\Psi_t : (\mathbf{x}, t) \rightarrow \Delta\mathbf{x}$. Without loss of generality, we chose $t = 0$ as the canonical scene $\Psi_t : (\mathbf{x}, 0) \rightarrow \mathbf{0}$. By doing so the scene is no longer independent between time instances, and becomes interconnected through a common canonical space anchor. Then, the assigned emitted color and volume density under viewing direction $\mathbf{d}$ equal to those in the canonical configuration $\Psi_x : (\mathbf{x} + \Delta\mathbf{x}, \mathbf{d}) \rightarrow (\mathbf{c}, \sigma)$.

We propose to learn Ψ_x and Ψ_t using a sparse set of T RGB images $\{\mathbf{I}_t, \mathbf{T}_t\}_{t=1}^{T}$ captured with a monocular camera, where $\mathbf{I}_t \in \mathbb{R}^{H \times W \times 3}$ denotes the image acquired under camera pose $\mathbf{T}_t \in \mathbb{R}^{4 \times 4}$ SE(3), at time t. Although we could assume multiple views per time instance, we want to test the limits of our method, and assume a single image per time instance. That is, we do not observe the scene under a specific configuration/deformation state from different viewpoints.

5.4.2 Implicit Model

We now introduce D-NeRF, our novel neural renderer for view synthesis trained solely from a sparse set of images of a dynamic scene. We build on NeRF [328] and generalize it to handle non-rigid scenes. Recall that NeRF requires multiple views of a rigid scene In contrast, D-NeRF can learn a volumetric density representation for continuous non-rigid scenes trained with a single view per time instant.

As shown in Fig. 5.12, D-NeRF consists of two main neural network modules, which parameterize the mappings explained in the previous subsection Ψ_t, Ψ_x. On the one hand we have the *Canonical Network*, an MLP (multilayer perceptron) $\Psi_x(\mathbf{x}, \mathbf{d}) \mapsto (\mathbf{c}, \sigma)$ is trained to encode

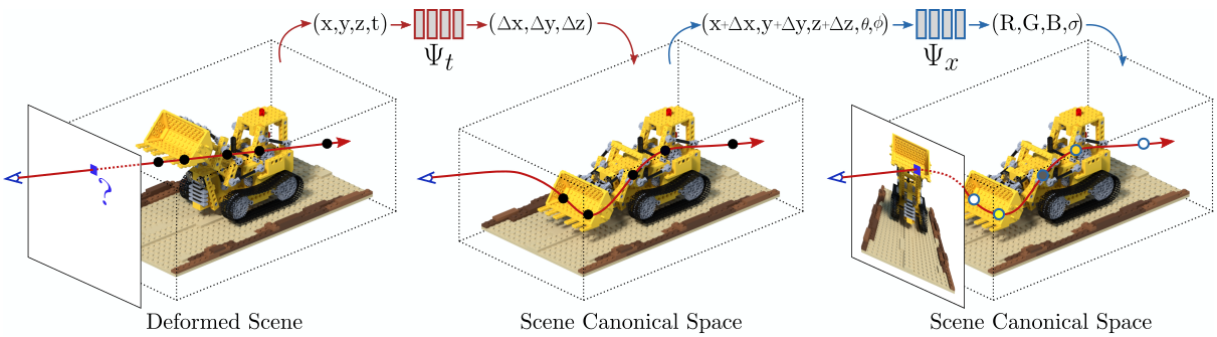


Figure 5.12: **D-NeRF model**. The proposed architecture consists of two main blocks: a deformation network Ψ_t mapping all scene deformations to a common canonical configuration; and a canonical network Ψ_x regressing volume density and view-dependent RGB color from every camera ray.

the scene in the canonical configuration such that given a 3D point $\mathbf{x}$ and a view direction $\mathbf{d}$ returns its emitted color $\mathbf{c}$ and volume density σ. The second module is called *Deformation Network* and consists of another MLP $\Psi_t(\mathbf{x}, t) \mapsto \Delta\mathbf{x}$ which predicts a deformation field defining the transformation between the scene at time t and the scene in its canonical configuration. We next describe in detail each one of these blocks, their interconnection for volume rendering and how are they learned.

Model Architecture

Canonical Network. With the use of a canonical configuration we seek to find a representation of the scene that brings together the information of all corresponding points in all images. By doing this, the missing information from a specific viewpoint can then be retrieved from that canonical configuration, which shall act as an anchor interconnecting all images.

The canonical network Ψ_x is trained so as to encode volumetric density and color of the scene in canonical configuration. Concretely, given the 3D coordinates $\mathbf{x}$ of a point, we first encode it into a 256-dimensional feature vector. This feature vector is then concatenated with the camera viewing direction $\mathbf{d}$, and propagated through a fully connected layer to yield the emitted color $\mathbf{c}$ and volume density σ for that given point in the canonical space.

Deformation Network. The deformation network Ψ_t is optimized to estimate the deformation field between the scene at a specific time instant and the scene in canonical space. Formally, given a 3D point $\mathbf{x}$ at time t, Ψ_t is trained to output the displacement $\Delta\mathbf{x}$ that transforms the given point to its position in the canonical space as $\mathbf{x} + \Delta\mathbf{x}$. For all experiments, without loss of

generality, we set the canonical scene to be the scene at time $t = 0$:

$$\Psi_t(\mathbf{x}, t) = \begin{cases} \Delta\mathbf{x}, & \text{if } t \neq 0 \\ 0, & \text{if } t = 0 \end{cases} \tag{5.15}$$

As shown in previous works [293, 328, 356], directly feeding raw coordinates and angles to a neural network results in low performance. Thus, for both the canonical and the deformation networks, we first encode $\mathbf{x}$, $\mathbf{d}$ and t into a higher dimension space. We use the same positional encoder as in [328] where $\gamma(p) =< (\sin(2^l\pi p), \cos(2^l\pi p)) >_0^L$. We independently apply the encoder $\gamma(\cdot)$ to each coordinate and camera view component, using $L = 10$ for $\mathbf{x}$, and $L = 4$ for $\mathbf{d}$ and t.

Volume Rendering

We now adapt NeRF volume rendering equations to account for non-rigid deformations in the proposed 6D neural radiance field. Let $\mathbf{x}(h) = \mathbf{o} + h\mathbf{d}$ be a point along the camera ray emitted from the center of projection $\mathbf{o}$ to a pixel p. Considering near and far bounds h_n and h_f in that ray, the expected color C of the pixel p at time t is given by:

$$C(p, t) = \int_{h_n}^{h_f} \mathcal{T}(h, t)\sigma(\mathbf{p}(h, t))\mathbf{c}(\mathbf{p}(h, t), \mathbf{d})dh, \tag{5.16}$$

$$\text{where} \quad \mathbf{p}(h, t) = \mathbf{x}(h) + \Psi_t(\mathbf{x}(h), t), \tag{5.17}$$

$$[\mathbf{c}(\mathbf{p}(h, t), \mathbf{d}), \sigma(\mathbf{p}(h, t))] = \Psi_x(\mathbf{p}(h, t), \mathbf{d}), \tag{5.18}$$

$$\text{and} \quad \mathcal{T}(h, t) = \exp\left(-\int_{h_n}^{h} \sigma(\mathbf{p}(s, t))ds\right). \tag{5.19}$$

The 3D point $\mathbf{p}(h, t)$ denotes the point on the camera ray $\mathbf{x}(h)$ transformed to canonical space using our Deformation Network Ψ_t, and $\mathcal{T}(h, t)$ is the accumulated probability that the ray emitted from h_n to h_f does not hit any other particle. Notice that the density σ and color $\mathbf{c}$ are predicted by our Canonical Network Ψ_x.

As in [328] the volume rendering integrals in Eq. (5.16) and Eq. (5.19) can be approximated via numerical quadrature. To select a random set of quadrature points $\{h_n\}_{n=1}^N \in [h_n, h_f]$ a stratified sampling strategy is applied by uniformly drawing samples from evenly-spaced ray

bins. A pixel color is approximated as:

$$C'(p,t) = \sum_{n=1}^{N} \mathcal{T}'(h_n,t)\alpha(h_n,t,\delta_n)\mathbf{c}(\mathbf{p}(h_n,t),\mathbf{d}), \tag{5.20}$$

$$\text{where} \quad \alpha(h,t,\delta) = 1 - \exp(-\sigma(\mathbf{p}(h,t))\delta), \tag{5.21}$$

$$\text{and} \quad \mathcal{T}'(h_n,t) = \exp\left(-\sum_{m=1}^{n-1} \sigma(\mathbf{p}(h_m,t))\delta_m\right), \tag{5.22}$$

and $\delta_n = h_{n+1} - h_n$ is the distance between two quadrature points.

5.4.3 Learning the Model

The parameters of the canonical Ψ_x and deformation Ψ_t networks are simultaneously learned by minimizing the mean squared error with respect to the T RGB images $\{\mathbf{I}_t\}_{t=1}^{T}$ of the scene and their corresponding camera pose matrices $\{\mathbf{T}_t\}_{t=1}^{T}$. Recall that every time instant is only acquired by a single camera.

At each training batch, we first sample a random set of pixels $\{p_{t,i}\}_{i=1}^{N_s}$ corresponding to the rays cast from some camera position $\mathbf{T}_t$ to some pixels i of the corresponding RGB image t. We then estimate the colors of the chosen pixels using Eq. (5.20). The training loss we use is the mean squared error between the rendered and real pixels:

$$\mathcal{L} = \frac{1}{N_s}\sum_{i=1}^{N_s} \left\| \hat{C}(p,t) - C'(p,t) \right\|_2^2, \tag{5.23}$$

where $\hat{C}$ are the pixels' ground-truth color.

Implementation Details

Both the canonical network Ψ_x and the deformation network Ψ_t consists on simple 8-layers MLPs with ReLU activations. For the canonical network a final sigmoid non-linearity is applied to $\mathbf{c}$ and σ. No non-linearlity is applied to $\mathbf{\Delta x}$ in the deformation network.

For all experiments we set the canonical configuration as the scene state at $t = 0$ by enforcing it in Eq. (5.15). To improve the networks convergence, we sort the input images according to their time stamps (from lower to higher) and then we apply a curriculum learning strategy where we incrementally add images with higher time stamps.

The model is trained with 400×400 images during $800k$ iterations with a batch size of $N_s = 4096$ rays, each sampled 64 times along the ray. As for the optimizer, we use Adam [38] with learning rate of $5e-4$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and exponential decay to $5e-5$. The model is

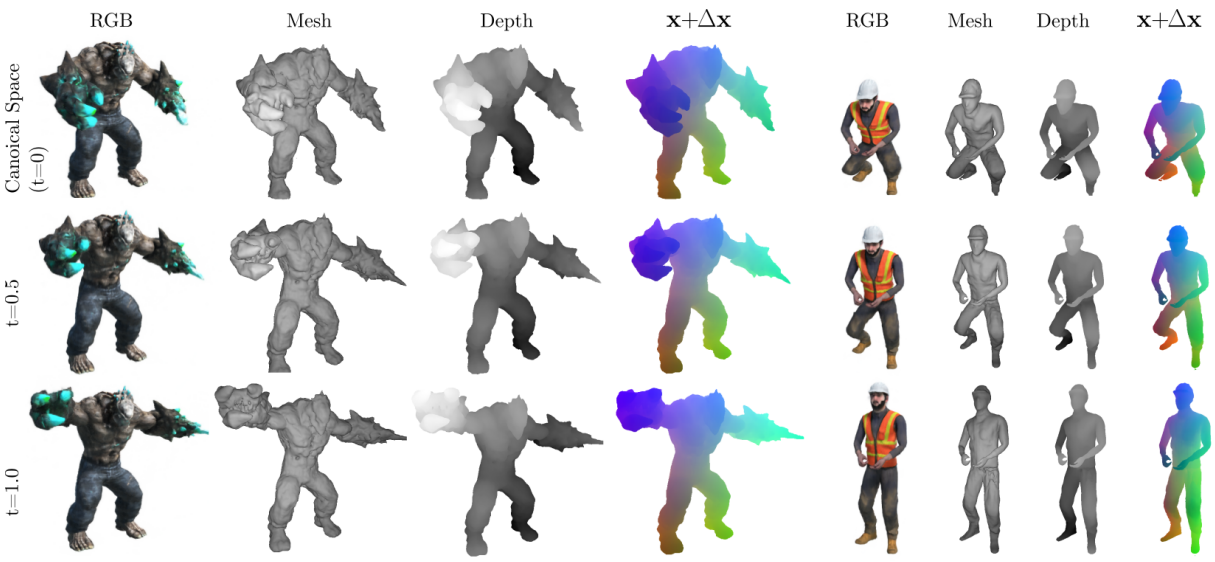


Figure 5.13: **Visualization of the learned scene representation.** Given a dynamic scene at a specific time instant, D-NeRF learns a displacement field $\Delta\mathbf{x}$ that maps all points $\mathbf{x}$ of the scene to a common canonical configuration. The volume density and view-dependent emitted radiance for this configuration is learned and transferred to the original input points to render novel views. This figure represents, from left to right: the learned radiance from a specific viewpoint, the volume density represented as a 3D mesh and a depth map, and the color-coded points of the canonical configuration mapped to the deformed meshes based on $\Delta\mathbf{x}$. The same colors on corresponding points indicate the correctness of such mapping.

trained with a single Nvidia® GTX 1080 for 2 days.

5.4.4 Experimental Evaluation

We next provide a thorough evaluation of our system. We first test the main components of the model, namely the canonical and deformation networks. We then compare D-NeRF against NeRF and T-NeRF, a variant in which does not use the canonical mapping. Finally, we demonstrate D-NeRF ability to synthesize novel views at an arbitrary time in several complex dynamic scenes.

In order to perform an exhaustive evaluation we have extended NeRF [328] rigid benchmark with eight scenes containing dynamic objects under large deformations and realistic non-Lambertian materials. As in the rigid benchmark of [328], six are rendered from viewpoints sampled from the upper hemisphere, and two are rendered from viewpoints sampled on the full sphere. Each scene contains between 100 and 200 rendered views depending on the action time span, all at 800×800 pixels. We released the path-traced images with defined train/validation/test splits for these eight scenes.

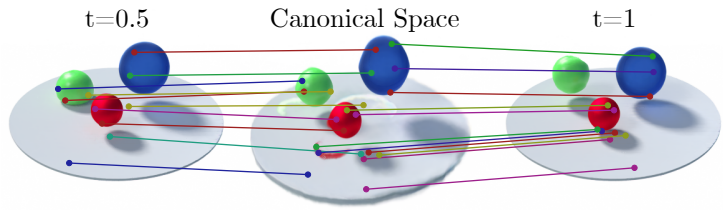


Figure 5.14: **Analyzing shading effects.** Pairs of corresponding points between the canonical space and the scene at times $t = 0.5$ and $t = 1$.

Dissecting the Model

This subsection provides insights about D-NeRF behaviour when modeling a dynamic scene and analyze the two main modules, namely the canonical and deformation networks.

We initially evaluate the ability of the canonical network to represent the scene in a canonical configuration. The results of this analysis for two scenes are shown the first row of Fig. 5.13 (columns 1-3 in each case). The plots show, for the canonical configuration ($t = 0$), the RGB image, the 3D occupancy network and the depth map, respectively. The rendered RGB image is the result of evaluating the canonical network on rays cast from an arbitrary camera position applying Eq. (5.20). To better visualize the learned volumetric density we transform it into a mesh applying marching cubes [357], with a 3D cube resolution of 256^3 voxels. Note how D-NeRF is able to model fine geometric and appearance details for complex topologies and texture patterns, even when it was only trained with a set of sparse images, each under a different deformation.

In a second experiment we assess the capacity of the network to estimate consistent deformation fields that map the canonical scene to the particular shape at each input image. The second and third rows of Fig. 5.13 show the result of applying the corresponding translation vectors to the canonical space for $t = 0.5$ and $t = 1$. The fourth column in each of the two examples visualizes the displacement field, where the color-coded points in the canonical shape ($t = 0$) at mapped to the different shape configurations at $t = 0.5$ and $t = 1$. Note that the colors are consistent along the time instants, indicating that the displacement field is correctly estimated.

Another question that we try to answer is how D-NeRF manages to model phenomena like shadows/shading effects, that is, how the model can encode changes of appearance of the same point along time. We have carried an additional experiment to answer this. In Fig. 5.14 we show a scene with three balls, made of very different materials (plastic –green–, translucent glass –blue– and metal –red–). The figure plots pairs of corresponding points between the canonical configuration and the scene at a specific time instant. D-NeRF is able to synthesize the shading

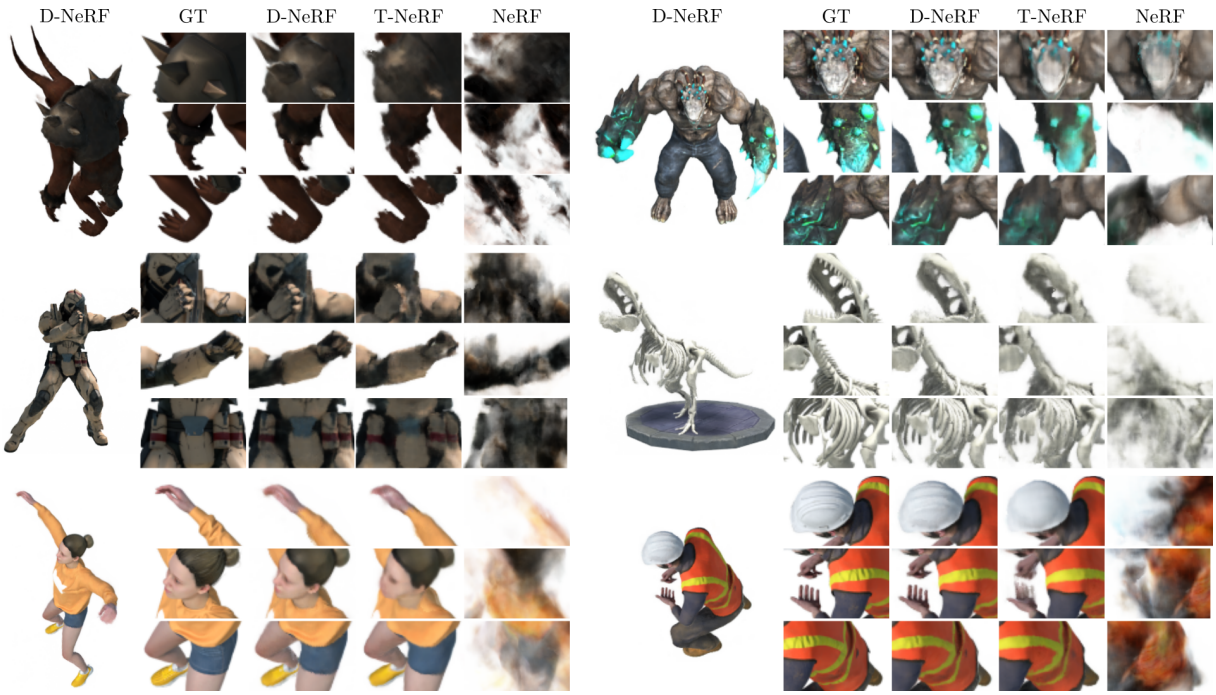


Figure 5.15: **Qualitative comparison.** Novel view synthesis results of dynamic scenes. For every scene we show an image synthesised from a novel view at an arbitrary time by our method, and three close-ups for: ground-truth, NeRF, T-NeRF, and D-NeRF (ours).

	Hell Warrior				Mutant				Hook				Bouncing Balls			
Method	MSE↓	PSNR↑	SSIM↑	LPIPS↓	MSE↓	PSNR↑	SSIM↑	LPIPS↓	MSE↓	PSNR↑	SSIM↑	LPIPS↓	MSE↓	PSNR↑	SSIM↑	LPIPS↓
NeRF	44e-3	13.52	0.81	0.25	9e-4	20.31	0.91	0.09	21e-3	16.65	0.84	0.19	1e-2	18.28	0.88	0.23
T-NeRF	47e-4	23.19	0.93	0.08	8e-4	30.56	0.96	0.04	18e-4	27.21	0.94	**0.06**	6e-4	32.01	0.97	0.04
D-NeRF	**31e-4**	**25.02**	**0.95**	**0.06**	**7e-4**	**31.29**	**0.97**	**0.02**	**11e-4**	**29.25**	**0.96**	0.11	**5e-4**	**32.80**	**0.98**	**0.03**
	Lego				T-Rex				Stand Up				Jumping Jacks			
Method	MSE↓	PSNR↑	SSIM↑	LPIPS↓	MSE↓	PSNR↑	SSIM↑	LPIPS↓	MSE↓	PSNR↑	SSIM↑	LPIPS↓	MSE↓	PSNR↑	SSIM↑	LPIPS↓
NeRF	9e-4	20.30	0.79	0.23	3e-3	24.49	0.93	0.13	1e-2	18.19	0.89	0.14	1e-2	18.28	0.88	0.23
T-NeRF	**3e-4**	**23.82**	**0.90**	**0.15**	9e-4	30.19	0.96	0.13	7e-4	31.24	0.97	0.02	6e-4	32.01	0.97	0.03
D-NeRF	6e-4	21.64	0.83	0.16	**6e-4**	**31.75**	**0.97**	**0.03**	**5e-4**	**32.79**	**0.98**	**0.02**	**5e-4**	**32.80**	**0.98**	**0.03**

Table 5.4: **Quantitative comparison.** We report MSE/LPIPS (lower is better) and PSNR/SSIM (higher is better).

effects by warping the canonical configuration. For instance, observe how the floor shadows are warped along time. Note that the points in the shadow of, *e.g.*the red ball, at $t = 0.5$ and $t = 1$ map at different regions of the canonical space.

Quantitative Comparison

We next evaluate the quality of D-NeRF on the novel view synthesis problem and compare it against the original NeRF [328], which represents the scene using a 5D input (x, y, z, θ, ϕ), and T-NeRF, a straight-forward extension of NeRF in which the scene is represented by a 6D input

$(x, y, z, \theta, \phi, t)$, without considering the intermediate canonical configuration of D-NeRF.

Table 5.4 summarizes the quantitative results on the 8 dynamic scenes of our dataset. We use several metrics for the evaluation: Mean Squared Error (MSE), Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM) [272] and Learned Perceptual Image Patch Similarity (LPIPS) [358]. In Fig. 5.15 we show samples of the estimated images under a novel view for visual inspection. As expected, NeRF is not able to model the dynamics scenes as it was designed for rigid cases, and always converges to a blurry mean representation of all deformations. On the other hand, the T-NeRF baseline is able to capture reasonably well the dynamics, although is not able to retrieve high frequency details. For example, in Fig. 5.15 top-left image it fails to encode the shoulder pad spikes, and in the top-right scene it is not able to model the stones and cracks. D-NeRF, instead, retains high details of the original image in the novel views. This is quite remarkable, considering that each deformation state has only been seen from a single viewpoint.

Additional Results

We finally show additional results to showcase the wide range of scenarios that can be handled with D-NeRF. Fig. 5.16 depicts, for four scenes, the images rendered at different time instants from two novel viewpoints. The first column displays the canonical configuration. Note that we are able to handle several types of dynamics: articulated motion in the *Tractor* scene; human motion in the *Jumping Jacks* and *Warrior* scenes; and asynchronous motion of several *Bouncing Balls*. Also note that the canonical configuration is a sharp and neat scene, in all cases, expect for the Jumping Jacks, where the two arms appear to be blurry. This, however, does not harm the quality of the rendered images, indicating that the network is able warp the canonical configuration so as to maximize the rendering quality. This is indeed consistent with Sec. 5.4.4 insights on how the network is able to encode shading.

5.5 Summary

In this chapter we have presented two novel methods for simultaneous reconstruction and synthesis.

For modeling the manifold of large 3D data corpus we have proposed C-Flow, a novel conditioning scheme for normalizing flows. This conditioning, in conjunction with a new strategy to model unordered 3D point clouds, has made it possible to address 3D reconstruction and rendering images from point clouds, problems which so far, could not be tackled with normalizing flows. Furthermore, we demonstrate C-Flow to be a general-purpose model, being

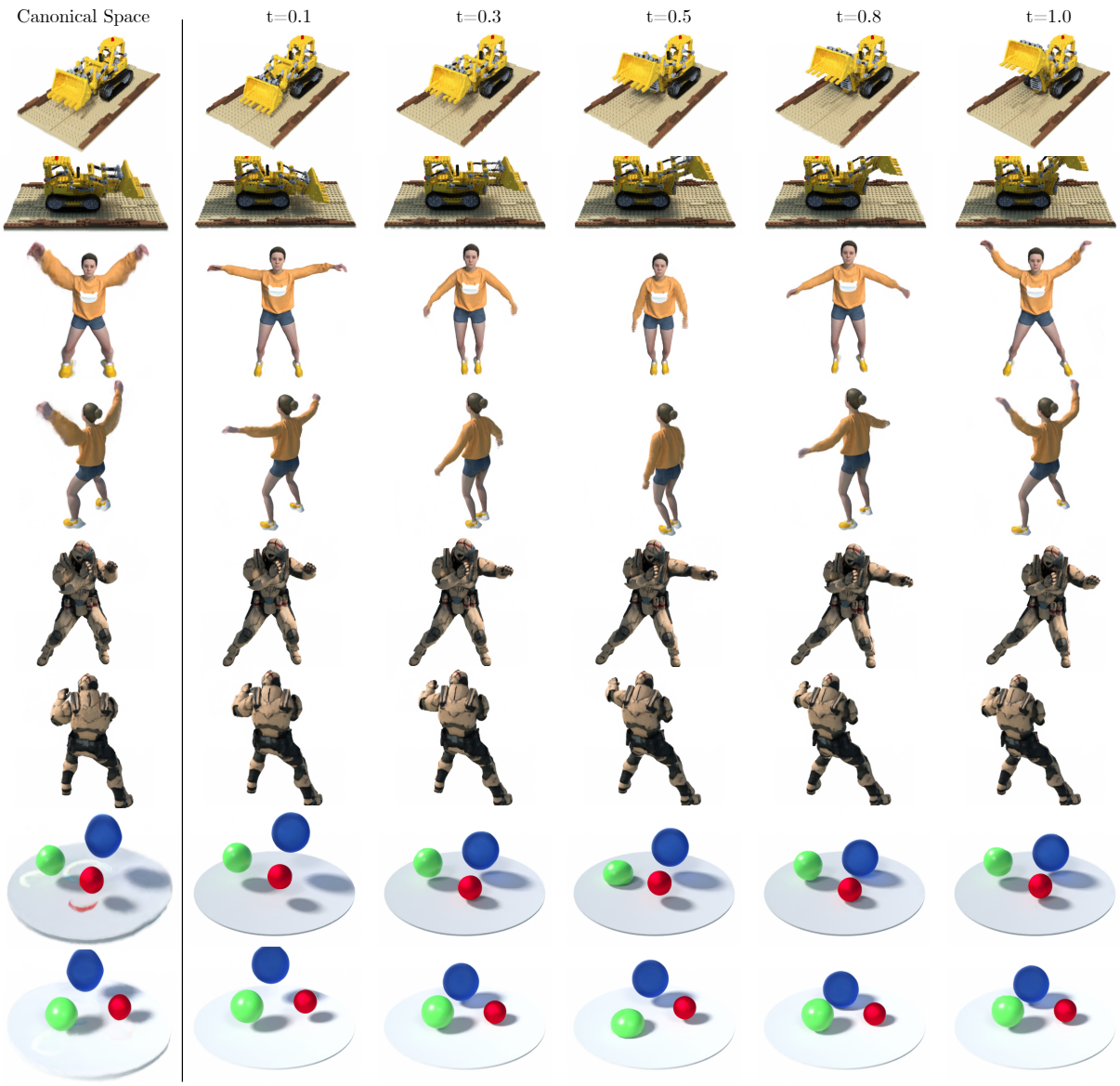


Figure 5.16: **Time & view conditioning.** Results of synthesising diverse scenes from two novel points of view across time and the learned canonical space. For every scene we also display the learned scene canonical space in the first column.

also applicable to many more multi-modality problems, such as image-to-image translation, style transfer and image content edition. To the best of our knowledge, no previous model has demonstrated such an adaptability.

For modeling a specific scene with extreme detail we have presented D-NeRF, a novel neural radiance field approach for modeling dynamic scenes. Our method can be trained end-to-end from only a sparse set of images acquired with a moving camera, and does not require pre-

computed 3D priors nor observing the same scene configuration from different viewpoints. The main idea behind D-NeRF is to represent time-varying deformations with two modules: one that learns a canonical configuration, and another that learns the displacement field of the scene at each time instant w.r.t. the canonical space. A thorough evaluation demonstrates that D-NeRF is able to synthesise high quality novel views of scenes undergoing different types of deformation, from articulated objects to human bodies performing complex body postures.

6
Conclusions

In this thesis we have addressed the problem of modeling the distribution of 3D objects for the tasks of monocular reconstruction and synthesis. We have devised a large set of techniques from low level data representations to full 3D pose estimation and rendering of rigid and non-rigid objects. And more importantly, we have finally bridged the gap between reconstruction and synthesis by proposing the first general-purpose model for bijective multi-modality transfer under the umbrella of flow-based generative models.

In Chapter 3, we defined a full pipeline for reconstructing a scene and the clothed persons in it while estimating the camera position. To do so, first, we presented a SLAM based algorithm for scene reconstruction and camera localization specifically designed to be robust to low textured scenarios by combining point and line correspondences. Next, to segment and track the elements in the scene, we devised the first real-time method for semi-supervised object segmentation and tracking from a single reference image. Once the elements are detected, we proposed what was the first deep network for 3D shape estimation of a non-rigid surface from a single image. This network was enhanced with novel geometric-aware layers embedding geometric properties as priors. We finally tackled the more complex task of clothed people reconstruction by proposing a fully-differentiable reconstruction algorithm and a novel 3D mesh representation. The results of our experimentation conclude that all proposed methods are robust to occlusions, self-occlusions and low-texture. In this chapter we also built two large-scale datasets to benchmark reconstruction algorithms that are publicly available.

In Chapter 4, we continued modeling 3D data, this time with a different approach. We introduced three generative methods to model the human body, an articulated nonrigid object. In experimentation we showed these methods to be capable of generating photo-realistic novel views of a person, face expressions and cloth transfer from a single image in the wild. All presented methods are unsupervised, that is, they do not require ground truth labels of the objective task. To circumvent the need of training data we proposed novel data representations,

architectures and losses capable of optimizing the model by only comparing the distributions of real and synthesized data. The work done in this chapter was awarded with the *Best Paper Award Honorable Mention* in ECCV 2018.

Finally, in Chapter 5, with all the knowledge acquired in previous chapters, we proposed an implicit based neural render and a bijective model for multi-modal transfer acting as an umbrella covering both reconstruction and synthesis into a single general-purpose model. To do so, we introduced a novel neural radiance field formulation and a conditioning scheme based on normalizing flows capable of explicitly modeling the conditional distribution of many domains including point clouds, images, segmentation masks and edge maps.

6.1 Future Work

Finally, we discuss some of the future research lines that arise from the work and contributions presented in this thesis that we have not yet fully exploited.

3D morphable models to support reconstruction. Part of this thesis focused on monocular reconstruction. While the presented methods yield very promising results, it would be interesting to tackle the problem in a coarse-to-fine manner with morphable models [138, 359]. First, estimate an initial rough estimation by regressing the parameters of a morphable model. And then, refine the initial estimate with a generative model. After finishing this thesis we have already started working in this direction. We are extending GimNet (Sec. 3.6) to run underneath a parametric representation of the human body wit the 3DMM SMPL [138].

Egocentric pose estimation. In Sec. 3.6, we tackled the task of 3D reconstruction of a person from am external point of view (*e.g.*picture of the person). It would be interesting to tackle such estimation with egocentric vision, which is, a first-person view captured by a wearable camera that approximates the visual field of the camera wearer. After finishing this thesis we have already started working in this direction. We have captured a large-scale egomotion dataset of 40 people doing 20 actions with egocentric videos. Each frame is annotated with the 3D skeleton joints. Using this dataset we intend to develop 3D pose estimation from egocentric videos.

Semantic information in reconstructed meshes. An interesting future line of research is to extend the reconstruction estimates with semantic information such as object category, material, texture and weight. We have started exploring this line of research in [360] (not included in this thesis) where we aim to predict human grasp affordances once the object has been reconstructed.

Geometry Image. Thorough this thesis we have explored novel data representations. In particular, in Sec. 3.6, we described and exploited geometry images. While the results obtained are very promising, there are still several avenues to explore. For instance, exploring new regularization schemes on the geometry images, as well as, reducing the amount of artifacts caused by the sphere projection step.

Context-aware estimation. The methods presented in this thesis exclusively focus on the specific object/person to be reconstructed/edited. We argue that taking into account the influence of the environment as a prior would be interesting to study. For instance, if a person is carrying a box, the configuration of the body arms and legs will be highly constrained by the 3D position of that box. Discovering such interrelations between the person and the object/s of the context (or another person he/she is interacting with), and how these interrelations constrain the body would dramatically improve performance. We have started exploring this line of research in [211] (not included in this thesis) to predict human motion given a sequence of past observations using a novel context-aware motion prediction architecture. We use a semantic-graph model where the nodes parameterize the human and objects in the scene and the learned edges represent their mutual interactions.

3D morphable models to support generative models. Regarding synthesis, we strongly believe that the direction to improve photo-realism and a avoid non-anthropomorphic errors is to embed 3D models of the underlying geometry in the neural networks. After finishing this thesis we have already started working in this direction. We are extending GANimation (Sec. 4.4) to run underneath the 3DMM [359] representing the human face.

Flow-based models. Flow models have highly desirable properties like exact log-likelihood evaluation and exact latent-variable inference, however they are still in their infancy and have not received as much attention as alternative generative models. In Chapter 5, we introduced a novel conditioning scheme that brings normalizing flows to an entirely new scenario with great possibilities for multi-modal data modeling. However, for flows to become popular there is still one key piece missing, lightweight models. Current models are too computational demanding, requiring large data centers to train. Out of all mention future research lines, we believe making flow-based models lighter is the one with greatest potential impact on the research community.

A

List of Publications & Awards

In this section, the reader can find the complete list of publications and awards derived from this thesis. **Only first author works were included in this dissertation.**

Book Chapters

1. **A. Pumarola**, A. Vakhitov, A. Agudo, F. Moreno-Noguer, and A. Sanfeliu. "Relative localization for aerial manipulation with PL-SLAM." In Aerial Robotic Manipulation, pp. 239-248, 2019.

2. E. Guerra, **A. Pumarola**, A. Grau, A. Sanfeliu. "Perception for Detection and Grasping." In Aerial Robotic Manipulation, pp. 275-283, 2019.

Journals

3. **A. Pumarola**, A. Agudo, A. M. Martinez, A. Sanfeliu, and F. Moreno-Noguer. "GANimation: One-shot anatomically consistent facial animation." In International Journal of Computer Vision (IJCV), pp. 698-713, 2019.

4. **A. Pumarola**, J. Sanchez-Riera, G. Choi, A. Sanfeliu, and F. Moreno-Noguer. "3DPeople: Modeling the geometry of dressed humans." In International Journal of Computer Vision (IJCV), Under Review, 2020.

5. **A. Pumarola**, V. Goswami, F. Vicente, F. De la Torre, F. Moreno-Noguer. "Unsupervised Image-to-Video Clothing Transfer." In International Journal of Computer Vision (IJCV), Under Review, 2020.

Conferences

6. **A. Pumarola**, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer. "PL-SLAM: Real-time monocular visual SLAM with points and lines." In IEEE international conference on robotics and automation (ICRA), pp. 4503-4508, 2017.

7. **A. Pumarola**, A. Agudo, L. Porzi, A. Sanfeliu, V. Lepetit, and F. Moreno-Noguer. "Geometry-aware network for non-rigid shape prediction from a single view." In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4681-4690, 2018.

8. **A. Pumarola**, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer. "Unsupervised person image synthesis in arbitrary poses." In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8620-8628, 2018. **Spotlight (top 6.7%)**.

9. **A. Pumarola**, A. Agudo, A. M. Martinez, A. Sanfeliu, and F. Moreno-Noguer. "GANimation: Anatomically-aware facial animation from a single image." In European Conference on Computer Vision (ECCV), pp. 818-833, 2018. **Best Paper Award Honorable Mention**.

10. **A. Pumarola**, J. Sanchez-Riera, G. Choi, A. Sanfeliu, and F. Moreno-Noguer. "3DPeople: Modeling the geometry of dressed humans." In IEEE International Conference on Computer Vision (ICCV), pp. 2242-2251, 2019.

11. E. Corona, **A. Pumarola**, G. Alenyà, F. Moreno-Noguer. "Context-aware Human Motion Prediction." In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6992-7001, 2020.

12. E. Corona, **A. Pumarola**, G. Alenyà, F. Moreno-Noguer, G. Rogez. "GanHand: Predicting Human Grasp Affordances in Multi-Object Scenes." In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5031-5041, 2020.

13. **A. Pumarola**, S. Popov, F. Moreno-Noguer, V. Ferrari. "C-Flow: Conditional Generative Flow Models for Images and 3D Point Clouds." In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7949-7958, 2020.

14. **A. Pumarola**, E. Corona, G. Pons-Moll, F. Moreno-Noguer. "D-NeRF: Neural Radiance Fields for Dynamic Scenes." In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 10318-10327, 2021.

15. E. Corona, **A. Pumarola**, G. Alenya, G. Pons-Moll, F. Moreno-Noguer. "SMPLicit: Topology-aware Generative Model for Clothed People." In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 11875-11885, 2021.

Workshops

16. **A. Pumarola**, V. Goswami, F. Vicente, F. De la Torre, F. Moreno-Noguer " Unsupervised Image-to-Video Clothing Transfer." In IEEE International Conference on Computer Vision Workshops (ICCV). 2019

ArXiv

17. S. Athar, **A. Pumarola**, F. Moreno-Noguer, D. Samaras. "FaceDet3D: Facial Expressions with 3D Geometric Detail Prediction.", arXiv, 2021.

18. S. Caelles*, **A. Pumarola***, F. Moreno-Noguer, A. Sanfeliu, and L. V. Gool. "Fast video object segmentation with Spatio-Temporal GANs.", arXiv, 2019.

Awards

- Google Research Award – Geometry-aware CNNs for Non-Rigid Reconstruction, 2017
- Amazon Research Award – Geometry-aware 3D Human Body Animation from Still Photos, 2018
- ECCV Best Paper Award Honorable Mention – GANimation: Anatomically-aware facial animation from a single image, 2018
- Google Research Award – GANimation3D: Unsupervised 3D Face Animation from Monocular Images, 2019

Bibliography

[1] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: a versatile and accurate monocular slam system," *T-RO*, 2015.

[2] A. Varol, M. Salzmann, P. Fua, and R. Urtasun, "A constrained latent variable model," in *CVPR*, 2012.

[3] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid, "Learning from synthetic humans," in *CVPR*, 2017.

[4] P. T. Choi, K. C. Lam, and L. M. Lui, "Flash: Fast landmark aligned spherical harmonic parameterization for genus-0 closed brain surfaces," *SIIMS*, 2015.

[5] A. Sinha, J. Bai, and K. Ramani, "Deep learning 3D Shape Surfaces using Geometry Images," in *ECCV*, 2016.

[6] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik, "End-to-end Recovery of Human Shape and Pose," in *CVPR*, 2018.

[7] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang, "Deepfashion: Powering robust clothes recognition and retrieval with rich annotations," in *CVPR*, 2016.

[8] M. Li, W. Zuo, and D. Zhang, "Deep identity-aware transfer of facial attributes," *arXiv preprint arXiv:1610.05586*, 2016.

[9] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *ICLR*, 2015.

[10] G. Perarnau, J. van de Weijer, B. Raducanu, and J. M. Álvarez, "Invertible conditional GANs for image editing," *arXiv preprint arXiv:1611.06355*, 2016.

[11] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," *CVPR*, 2018.

[12] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," in *NeurIPS*, 2018.

[13] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *IROS*.

[14] B. Zhao, X. Wu, Z.-Q. Cheng, H. Liu, and J. Feng, "Multi-view image generation from a single-view," *arXiv preprint arXiv:1704.04886*, 2017.

[15] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz, "Mocogan: Decomposing motion and content for video generation," *CVPR*, 2018.

[16] T. Salimans, I. Goodfellow, W. Zaremba, V.Cheung, A. .Radford, and X. Chen, "Improved techniques for training gans," in *NeurIPS*, 2016.

[17] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," *CVPR*, 2019.

[18] A. Pumarola, A. Agudo, A. M. Martinez, A. Sanfeliu, and F. Moreno-Noguer, "Ganimation: Anatomically-aware facial animation from a single image," in *ECCV*, 2018.

[19] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer, "Pl-slam: Real-time monocular visual slam with points and lines," in *ICRA*, 2017.

[20] A. Pumarola, A. Vakhitov, A. Agudo, F. Moreno-Noguer, and A. Sanfeliu, "Relative localization for aerial manipulation with pl-slam," in *Aerial Robotic Manipulation*, 2019.

[21] E. Guerra, A. Pumarola, A. Grau, and A. Sanfeliu, "Perception for detection and grasping," in *Aerial Robotic Manipulation*, 2019.

[22] S. Caelles, A. Pumarola, F. Moreno-Noguer, A. Sanfeliu, and L. Van Gool, "Fast video object segmentation with spatio-temporal gans," *arXiv preprint arXiv:1903.12161*, 2019.

[23] A. Pumarola, A. Agudo, L. Porzi, A. Sanfeliu, V. Lepetit, and F. Moreno-Noguer, "Geometry-aware network for non-rigid shape prediction from a single view," in *CVPR*, 2018.

[24] A. Pumarola, J. Sanchez, G. Choi, A. Sanfeliu, and F. Moreno-Noguer, "3dpeople: Modeling the geometry of dressed humans," in *ICCV*, 2019.

[25] A. Pumarola, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer, "Unsupervised person image synthesis in arbitrary poses," in *CVPR*, 2018.

[26] A. Pumarola, A. Agudo, A. M. Martinez, A. Sanfeliu, and F. Moreno-Noguer, "Ganimation: One-shot anatomically consistent facial animation," *IJCV*, 2019.

[27] A. Pumarola, V. Goswami, F. Vicente, F. De la Torre, and F. Moreno-Noguer, "Unsupervised image-to-video clothing transfer," in *ICCV Workshops*, 2019.

[28] A. Pumarola, S. Popov, F. Moreno-Noguer, and V. Ferrari, "C-flow: Conditional generative flow models for images and 3d point clouds," in *CVPR*, 2020.

[29] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer, "D-nerf: Neural radiance fields for dynamic scenes," in *CVPR*, 2021.

[30] C. Cortes and V. Vapnik, "Support-vector networks," *JMLR*, 1995.

[31] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[32] D. Michie, D. Spiegelhalter, and C. Taylor, "Machine learning, neural and statistical classification," 1994.

[33] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.

[34] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, 1988.

[35] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," in *Shape, Contour and Grouping in Computer Vision*, Springer-Verlag, 1999.

[36] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *CVPR*, 2009.

[37] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *NeurIPS*, 2012.

[38] D. Kingma and J. Ba, "ADAM: A method for stochastic optimization," in *ICLR*, 2015.

[39] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, 2014.

[40] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NeurIPS*, 2014.

[41] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," in *ICLR*, 2014.

[42] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *ICML*, 2015.

[43] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *ICML*, 2017.

[44] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein GANs," in *NeurIPS*, 2017.

[45] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee., "Generative adversarial text to image synthesis," in *ICML*, 2016.

[46] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *ICCV*, 2017.

[47] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[48] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *ICML*, 2017.

[49] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *CVPR*, 2017.

[50] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," in *ICLR*, 2016.

[51] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *CVPR*, 2016.

[52] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CVPR*, 2017.

[53] A. Davison, I. Reid, N. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *TPAMI*, 2007.

[54] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *ISMAR*, 2007.

[55] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, "Generic and real-time structure from motion using local bundle adjustment," *Image and Vision Computing*, 2009.

[56] H. Strasdat, J. Montiel, and A. Davison, "Visual SLAM: Why Filter?," *Image and Vision Computing*, 2012.

[57] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *ECCV*, 2006.

[58] G. Klein and D. Murray, "Improving the agility of keyframe-based SLAM," in *ECCV*, 2008.

[59] G. Sibley, C. Mei, I. Reid, and P. Newman, "Adaptive relative bundle adjustment," in *Robotics: Science and Systems Conference*, 2009.

[60] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *ECCV*, 2014.

[61] R. Newcome and A. J. Davison, "Live dense reconstruction with a single moving camera," in *CVPR*, 2010.

[62] M. Pizzoli, C. Forster, and D. Scaramuzza, "REMODE: Probabilistic, monocular dense reconstruction in real time," in *ICRA*, 2014.

[63] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO 2.0: Semi-direct visual odometry for monocular and multicamera systems," *T-RO*, 2016.

[64] S. Bao, M. Bagra, Y. Chao, and S. Savarese, "Semantic structure from motion with points, regions, and objects," in *CVPR*, 2012.

[65] A. Concha, W. Hussain, L. Montano, and J. Civera, "Incorporating scene priors to dense monocular mapping," *Autonomous Robots*, 2015.

[66] J. Sola, T. Vidal-Calleja, J. Civera, and J. Montiel, "Impact of landmark parametrization on monocular EKF-SLAM with points and lines," *IJCV*, 2012.

[67] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," in *CVPR*, 2016.

[68] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool, "The 2017 DAVIS challenge on video object segmentation," *arXiv:1704.00675*, 2017.

[69] N. Xu, L. Yang, Y. Fan, D. Yue, Y. Liang, J. Yang, and T. Huang, "YouTube-VOS: A large-scale video object segmentation benchmark," *arXiv preprint arXiv:1809.03327*, 2018.

[70] J. Chang, D. Wei, and J. W. Fisher III, "A video representation using temporal superpixels," in *CVPR*, 2013.

[71] M. Grundmann, V. Kwatra, M. Han, and I. A. Essa, "Efficient hierarchical graph-based video segmentation," in *CVPR*, 2010.

[72] N. Märki, F. Perazzi, O. Wang, and A. Sorkine-Hornung, "Bilateral space video segmentation," in *CVPR*, 2016.

[73] F. Perazzi, O. Wang, M. Gross, and A. Sorkine-Hornung, "Fully connected object proposals for video segmentation," in *ICCV*, 2015.

[74] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool, "One-shot video object segmentation," in *CVPR*, 2017.

[75] F. Perazzi, A. Khoreva, R. Benenson, B. Schiele, and A. Sorkine-Hornung, "Learning video object segmentation from static images," in *CVPR*, 2017.

[76] P. Voigtlaender and B. Leibe, "Online adaptation of convolutional neural networks for video object segmentation," in *BMVC*, 2017.

[77] K.-K. Maninis, S. Caelles, Y. Chen, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool, "Video object segmentation without temporal information," *TPAMI*, 2018.

[78] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *ICCV*, 2017.

[79] J. Shin Yoon, F. Rameau, J. Kim, S. Lee, S. Shin, and I. So Kweon, "Pixel-level matching for video object segmentation using convolutional neural networks," in *ICCV*, 2017.

[80] L. Yang, Y. Wang, X. Xiong, J. Yang, and A. K. Katsaggelos, "Efficient video object segmentation via network modulation," in *CVPR*, 2018.

[81] S. Wug Oh, J.-Y. Lee, K. Sunkavalli, and S. Joo Kim, "Fast video object segmentation by reference-guided mask propagation," in *CVPR*, 2018.

[82] Y.-T. Hu, J.-B. Huang, and A. Schwing, "Maskrnn: Instance level video object segmentation," in *NeurIPS*, 2017.

[83] Y. Chen, J. Pont-Tuset, A. Montes, and L. Van Gool, "Blazingly fast video object segmentation with pixel-wise metric learning," in *CVPR*, 2018.

[84] H. Ci, C. Wang, and Y. Wang, "Video object segmentation by learning location-sensitive embeddings," in *ECCV*, 2018.

[85] Y.-T. Hu, J.-B. Huang, and A. G. Schwing, "Videomatch: Matching based video object segmentation," in *ECCV*, 2018.

[86] P. Voigtlaender, Y. Chai, F. Schroff, H. Adam, B. Leibe, and L.-C. Chen, "Feelvos: Fast end-to-end embedding learning for video object segmentation," in *CVPR*, 2019.

[87] J. Cheng, Y.-H. Tsai, W.-C. Hung, S. Wang, and M.-H. Yang, "Fast and accurate online video object segmentation via tracking parts," in *CVPR*, 2018.

[88] L. Bao, B. Wu, and W. Liu, "Cnn in mrf: Video object segmentation via inference in a cnn-based higher-order spatio-temporal mrf," in *CVPR*, 2018.

[89] S. Chandra, C. Couprie, and I. Kokkinos, "Deep spatio-temporal random fields for efficient video segmentation," in *CVPR*, 2018.

[90] X. Li and C. Change Loy, "Video object segmentation with joint re-identification and attention-aware mask propagation," in *ECCV*, 2018.

[91] J. Cheng, Y.-H. Tsai, S. Wang, and M.-H. Yang, "Segflow: Joint learning for video object segmentation and optical flow," in *ICCV*, 2017.

[92] S. Jain, B. Xiong, and K. Grauman, "Fusionseg: Learning to combine motion and appearance for fully automatic segmention of generic objects in videos," *CVPR*, 2017.

[93] P. Hu, G. Wang, X. Kong, J. Kuen, and Y.-P. Tan, "Motion-guided cascaded refinement network for video object segmentation," in *CVPR*, 2018.

[94] H. Xiao, J. Feng, G. Lin, Y. Liu, and M. Zhang, "Monet: Deep motion exploitation for video object segmentation," in *CVPR*, 2018.

[95] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, 1997.

[96] C. Ventura, M. Bellver, A. Girbau, A. Salvador, F. Marques, and X. Giro-i Nieto, "Rvos: End-to-end recurrent network for video object segmentation," in *CVPR*, 2019.

[97] P. Luc, C. Couprie, S. Chintala, and J. Verbeek, "Semantic segmentation using adversarial networks," in *NeurIPS Workshop on Adversarial Training*, 2016.

[98] N. Souly, C. Spampinato, and M. Shah, "Semi and weakly supervised semantic segmentation using generative adversarial network," in *ICCV*, 2017.

[99] Y. Luo, Z. Zheng, L. Zheng, T. Guan, J. Yu, and Y. Yang, "Macro-micro adversarial network for human parsing," in *ECCV*, 2018.

[100] W.-C. Hung, Y.-H. Tsai, Y.-T. Liou, Y.-Y. Lin, and M.-H. Yang, "Adversarial learning for semi-supervised semantic segmentation," in *BMVC*, 2018.

[101] D. Metaxas and D. Terzopoulos, "Constrained deformable superquadrics and nonrigid motion tracking," in *CVPR*, 1991.

[102] T. McInerney and D. Terzopoulos, "A finite element model for 3D shape reconstruction and nonrigid motion tracking," in *ICCV*, 1993.

[103] Y. Kita, "Elastic-model driven analysis of several views of a deformable cylindrical object," *TPAMI*, 1996.

[104] T. McInerney and D. Terzopoulos, "A dynamic finite element surface model for segmentation and tracking in multidimensional medical images with application to cardiac 4D image analysis," *CMIG*, 1995.

[105] A. Bartoli, Y. Gérard, F. Chadebecq, T. Collins, and D. Pizarro, "Shape-from-template," *TPAMI*, 2015.

[106] A. Chhatkuli, D. Pizzaro, and A. Bartoli, "Stable template-based isometric 3d reconstruction in all imaging conditions by linear least-squares," in *CVPR*, 2014.

[107] F. Moreno-Noguer and P. Fua, "Stochastic exploration of ambiguities for nonrigid shape recovery," *TPAMI*, 2013.

[108] F. Moreno-Noguer, M. Salzmann, V. Lepetit, and P. Fua, "Capturing 3D stretchable surfaces from single images in closed form," in *CVPR*, 2009.

[109] J. Östlund, A. Varol, D. T. Ngo, and P. Fua, "Laplacian meshes for monocular 3D shape recovery," in *ECCV*, 2012.

[110] M. Perriollat, R. Hartley, and A. Bartoli, "Monocular template-based reconstruction of inextensible surfaces," *IJCV*, 2011.

[111] M. Salzmann and P. Fua, "Reconstructing sharply folding surfaces: A convex formulation," in *CVPR*, 2009.

[112] M. Salzmann, F. Moreno-Noguer, V. Lepetit, and P. Fua, "Closed-form solution to non-rigid 3D surface registration," *ECCV*, 2008.

[113] S. Vicente and L. Agapito, "Soft inextensibility constraints for template-free non-rigid reconstruction," *ECCV*, 2012.

[114] J. Sanchez, J. Östlund, P. Fua, and F. Moreno-Noguer, "Simultaneous pose, correspondence and non-rigid shape," in *CVPR*, 2010.

[115] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method," *IJCV*, 1992.

[116] A. Agudo and F. Moreno-Noguer, "Simultaneous pose and non-rigid shape with particle dynamics," in *CVPR*, 2015.

[117] A. Agudo, J. M. M. Montiel, L. Agapito, and B. Calvo, "Modal space: A physics-based model for sequential estimation of time-varying shape from monocular video," *JMIV*, 2017.

[118] A. Agudo and F. Moreno-Noguer, "DUST: Dual union of spatio-temporal subspaces for monocular multiple object 3D reconstruction," in *CVPR*, 2017.

[119] A. Agudo, F. Moreno-Noguer, B. Calvo, and J. Montiel, "Real-time 3d reconstruction of non-rigid shapes with a single moving camera," *CVIU*, vol. 153, pp. 37–54, December 2016.

[120] M. Lee, C. H. Choi, and S. Oh, "A procrustean markov process for non-rigid structure recovery," in *CVPR*, 2014.

[121] L. Torresani, A. Hertzmann, and C. Bregler, "Nonrigid structure-from-motion: estimating shape and motion with hierarchical priors," *TPAMI*, 2008.

[122] I. Akhter, Y. Sheikh, S. Khan, and T. Kanade, "Trajectory space: A dual representation for nonrigid structure from motion," *TPAMI*, 2011.

[123] A. Agudo and F. Moreno-Noguer, "Learning shape, motion and elastic models in force space," in *ICCV*, 2015.

[124] E. Trulls, S. Tsogkas, I. Kokkinos, A. Sanfeliu, and F. Moreno-Noguer, "Segmentation-aware deformable part models," in *CVPR*, pp. 168–175, 2014.

[125] E. Simo-Serra, C. Torras, and F. Moreno-Noguer, "DaLI: Deformation and light invariant descriptor," *IJCV*, vol. 115, no. 2, pp. 135–154, 2015.

[126] F. Moreno-Noguer, "Deformation and illumination invariant feature point descriptor," in *CVPR*, 2011.

[127] E. Trulls, , A. Sanfeliu, and F. Moreno-Noguer, "Spatiotemporal descriptor for wide-baseline stereo reconstruction of non-rigid and ambiguous scenes," in *ECCV*, 2012.

[128] A. Ramisa, G. Alenya, F. Moreno-Noguer, and C. Torras, "Learning rgb-d descriptors of garment parts for informed grasping," vol. 35, pp. 246–258, October 2014.

[129] A. O. Balan, M. J. Black, H. Haussecker, and L. Sigal, "Shining a light on human pose: On shadows, shading and the estimation of pose and shape," in *ICCV*, 2007.

[130] M. de La Gorce, N. Paragios, and D. J. Fleet, "Model-based hand tracking with texture, shading and self-occlusions," in *CVPR*, 2008.

[131] X. Wang, M. Salzmann, F. Wang, and J. Zhao, "Template-free 3D reconstruction of poorly-textured nonrigid surfaces," in *ECCV*, 2016.

[132] R. White and D. A. Forsyth, "Combining cues: Shape from shading and texture," in *CVPR*, 2006.

[133] T. F. Cootes, G. J. Edwards, and C. J. Taylor, "Active appearance models," *TPAMI*, 2001.

[134] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3D faces," in *SIGGRAPH*, 1999.

[135] I. Matthews and S. Baker, "Active appearance models revisited," *IJCV*, 2004.

[136] F. Moreno-Noguer and J. M. Porta, "Probabilistic simultaneous pose and non-rigid shape recovery," in *CVPR*, 2011.

[137] M. Salzmann, R. Urtasun, and P. Fua, "Local deformation models for monocular 3D shape recovery," in *CVPR*, 2008.

[138] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "SMPL: A skinned multi-person linear model," *TOG*, 2015.

[139] A. O. Balan, L. Sigal, M. J. Black, J. E. Davis, and H. W. Haussecker, "Detailed human shape and pose from images," in *CVPR*, 2007.

[140] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black, "Keep it SMPL: Automatic Estimation of 3D Human Pose and Shape from a Single Image," in *ECCV*, 2016.

[141] C. Lassner, J. Romero, M. Kiefel, F. Bogo, M. J. Black, and P. V. Gehler, "Unite the People: Closing the Loop between 3D and 2D human representations," in *CVPR*, 2017.

[142] E. Dibra, H. Jain, C. Öztireli, R. Ziegler, and M. Gross, "Human Shape from Silhouettes using Generative HKS Descriptors and Cross-modal Neural Networks," in *CVPR*, 2017.

[143] V. Tan, I. Budvytis, and R. Cipolla, "Indirect Deep structured Learning for 3D Human Body Shape and Pose Prediction," in *BMVC*, 2017.

[144] H.-Y. Tung, H.-W. Tung, E. Yumer, and K. Fragkiadaki, "Self-supervised learning of motion capture," in *NeurIPS*, 2017.

[145] T. Alldieck, M. Magnor, B. L. Bhatnagar, C. Theobalt, and G. Pons-Moll, "Learning to reconstruct people in clothing from a single RGB camera," in *CVPR*, 2019.

[146] T. Alldieck, M. Magnor, W. Xu, C. Theobalt, and G. Pons-Moll, "Video based reconstruction of 3d people models," in *CVPR*, 2018.

[147] T. Alldieck, M. Magnor, W. Xu, C. Theobalt, and G. Pons-Moll, "Detailed human avatars from monocular video," in *3DV*, 2018.

[148] V. Lazova, E. Insafutdinov, and G. Pons-Moll, "360-degree textures of people in clothing from a single image," in *3DV*, 2019.

[149] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li, "Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization," in *ICCV*, 2019.

[150] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta, "Learning a predictable and generative vector representation for objects," in *ECCV*, 2016.

[151] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee, "Perspective Transformer Nets: Learning Single-view 3D object Reconstruction without 3D Supervision," in *NeurIPS*, 2016.

[152] M. Tatarchenko, A. Dosovitskiy, and T. Brox, "Octree Generating Networks: Efficient Convolutional Architectures for High-resolution 3D Outputs," in *ICCV*, 2017.

[153] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, "O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis," *TOG*, 2017.

[154] P.-S. Wang, C.-Y. Sun, Y. Liu, and X. Tong, "Adaptive O-CNN: A Patch-based Deep Representation of 3D Shapes," *TOG*, 2018.

[155] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik, "A Point Set Generation Network for 3D Object Reconstruction from a Single Image.," in *CVPR*, 2017.

[156] A. Sinha, A. Unmesh, Q. Huang, and K. Ramani, "SurfNet: Generating 3D shape surfaces using deep residual networks," in *CVPR*, 2017.

[157] X. Gu, S. J. Gortler, and H. Hoppe, "Geometry images," in *TOG*, ACM, 2002.

[158] G. Varol, D. Ceylan, B. Russell, J. Yang, E. Yumer, I. Laptev, and C. Schmid, "BodyNet: Volumetric inference of 3D human body shapes," in *ECCV*, 2018.

[159] R. Natsume, S. Saito, Z. Huang, W. Chen, C. Ma, H. Li, and S. Morishima, "SiCloPe: Silhouette-Based Clothed People," in *CVPR*, 2019.

[160] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *T-RO*, 2017.

[161] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-D mapping with an RGB-D camera," *T-RO*, 2014.

[162] N. Ayache and O. D. Faugeras, "Building, registrating, and fusing noisy visual maps," *IJRR*, 1988.

[163] W. Y. Jeong and K. M. Lee, "Visual SLAM with line and corner features," in *IROS*, 2006.

[164] P. Newman, J. Leonard, J. D. Tardós, and J. Neira, "Explore and return: Experimental validation of real-time concurrent mapping and localization," in *ICRA*, 2002.

[165] A. J. D. P. Smith, I. D. Reid, "Real-time monocular SLAM with straight lines.," in *BMVC*, 2006.

[166] A. Vakhitov, J. Funke, and F. Moreno-Noguer, "Accurate and linear time pose estimation from points and lines," in *ECCV*, 2016.

[167] O. D. Faugeras and F. Lustman, "Motion and structure from motion in a piecewise planar environment," *IJPRAI*, 1988.

[168] W. Tan, H. Liu, Z. Dong, G. Zhang, and H. Bao, "Robust monocular SLAM in dynamic environments," in *ISMAR*, 2013.

[169] R. G. von Gioi, J. Jakubowicz, J. M. Morel, and G. Randall, "LSD: a line segment detector," *IPOL*, 2012.

[170] L. Zhang and R. Koch, "An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency," *JVCIR*, 2013.

[171] F. Moreno-Noguer, V. Lepetit, and P. Fua, "Accurate non-iterative o (n) solution to the pnp problem," in *ICCV*, 2007.

[172] Z. Kukelova, M. Bujnak, and T. Pajdla, "Polynomial eigenvalue solutions to minimal problems in computer vision," *TPAMI*, 2012.

[173] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second ed., 2004.

[174] J. Luiten, P. Voigtlaender, and B. Leibe, "Premvos: Proposal-generation, refinement and merging for video object segmentation," in *ACCV*, 2018.

[175] D. Tsai, M. Flagg, A. Nakazawa, and J. M. Rehg, "Motion coherent tracking using multi-label MRF optimization," *IJCV*, 2012.

[176] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun, "Large kernel matters–improve semantic segmentation by global convolutional network," in *CVPR*, 2017.

[177] S. Xie and Z. Tu, "Holistically-nested edge detection," in *ICCV*, 2015.

[178] C. Villani, *Optimal transport: old and new*. Springer Science & Business Media, 2008.

[179] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[180] K. Xu, L. Wen, G. Li, L. Bo, and Q. Huang, "Spatiotemporal cnn for video object segmentation," in *CVPR*, 2019.

[181] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," *ECCV*, 2014.

[182] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *IJCV*, 2010.

[183] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," *ICCV*, 2011.

[184] J. Shi, Q. Yan, L. Xu, and J. Jia, "Hierarchical image saliency detection on extended cssd," *TPAMI*, 2016.

[185] M.-M. Cheng, N. J. Mitra, X. Huang, P. H. Torr, and S.-M. Hu, "Global contrast based salient region detection," *TPAMI*, 2015.

[186] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *CVPR*, 2017.

[187] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *ICCV*, 2015.

[188] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *CVPR*, 2016.

[189] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *NeurIPS*, 2014.

[190] R. Garg, G. Carneiro, and I. Reid, "Unsupervised CNN for single view depth estimation: Geometry to the rescue," in *ECCV*, 2016.

[191] C. Godard, O. M. Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *CVPR*, 2017.

[192] D. Dai, H. Riemenschneider, and L. Van Gool, "The synthesizability of texture examples," in *CVPR*, 2014.

[193] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *ECCV*, 2016.

[194] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in *CVPR*, 2016.

[195] J. Martinez, R. Hossain, J. Romero, and J. J. Little, "A simple yet effective baseline for 3D human pose estimation," *ICCV*, 2017.

[196] F. Moreno-Noguer, "3D human pose estimation from a single image via distance matrix regression," in *CVPR*, 2017.

[197] E. A. Coutsias, C. Seok, and K. A. Dill, "Using quaternions to calculate rmsd," *JCC*, 2004.

[198] F. Brunet, R. Hartley, A. Bartoli, N. Navab, and R. Malgouyres, "Monocular template-based reconstruction of smooth and inextensible surfaces," in *ACCV*, 2010.

[199] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, 2004.

[200] N. Gumerov, A. Zandifar, R. Duraiswami, and L. S. Davis, "Structure of applicable surfaces from single views," in *ECCV*, 2004.

[201] S. Lieberknecht, S. Benhimane, P. Meier, and N. Navab, "A dataset and evaluation methodology for template-based tracking algorithms," in *ISMAR*, 2009.

[202] L. Sigal, A. O. Balan, and M. J. Black, "HumanEva: Synchronized Video and Motion Capture Dataset and Baseline Algorithm for Evaluation of Articulated Human Motion," *IJCV*, 2010.

[203] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, "Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments," *TPAMI*, 2014.

[204] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt, "VNect: Real-time 3D Human Pose Estimation with a Single RGB Camera," *TOG*, 2017.

[205] G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis, "Coarse-to-fine volumetric prediction for single-image 3D human pose," *arXiv preprint arXiv:1611.07828*, 2016.

[206] G. Rogez and C. Schmid, "MoCap-guided Data Augmentation for 3D Pose Estimation in the Wild," in *NeurIPS*, 2016.

[207] X. Sun, B. Xiao, S. Liang, and Y. Wei, "Integral Human Pose Regression," in *ECCV*, 2018.

[208] D. Tome, C. Russell, and L. Agapito, "Lifting from the deep: Convolutional 3D pose estimation from a single image," in *CVPR*, 2017.

[209] W. Yang, W. Ouyang, X. Wang, J. Ren, H. Li, and X. Wang, "3d human pose estimation in the wild by adversarial learning," in *CVPR*, 2018.

[210] A. Hernandez, J. Gall, and F. Moreno-Noguer, "Human motion prediction via spatio-temporal inpainting," in *ICCV*, 2019.

[211] E. Corona, A. Pumarola, G. Alenyà, and F. Moreno-Noguer, "Context-aware human motion prediction," in *CVPR*, 2020.

[212] E. Simo-Serra, C. Torras, and F. Moreno-Noguer, "3d human pose tracking priors using geodesic mixture models," *IJCV*, vol. 122, no. 2, 2017.

[213] P. Guan, A. Weiss, A. O. Balan, and M. Black, "Estimating Human Shape and Pose from a Single Image," in *ICCV*, 2009.

[214] A. Zanfir, E. Marinoiu, M. Zanfir, A.-I. Popa, and C. Sminchisescu, "Deep network for the integrated 3d sensing of multiple people in natural images," in *NeurIPS*, 2018.

[215] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis, "SCAPE: Shape Completion and Animation of People," *ACM*, 2005.

[216] S. Nadeem, Z. Su, W. Zeng, A. Kaufman, and X. Gu, "Spherical Parameterization Balancing Angle and Area Distortions," *TVCG*.

[217] B. Sapp and B. Taskar, "Modec: Multimodal Decomposable Models for Human Pose Estimation," in *CVPR*, 2013.

[218] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2D Human Pose Estimation: New Benchmark and State of the Art Analysis," in *CVPR*, 2014.

[219] S. Johnson and M. Everingham, "Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation," in *BMVC*, 2010.

[220] T. von Marcard, R. Henschel, M. Black, B. Rosenhahn, and G. Pons-Moll, "Recovering accurate 3d human pose in the wild using imus and a moving camera," in *ECCV*, 2018.

[221] https://www.adobe.com/es/products/fuse.html.

[222] http://www.makehumancommunity.org/.

[223] https://www.mixamo.com/.

[224] "Blender - a 3d modelling and rendering package.." https://www.blender.org/.

[225] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao, "LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop," *arXiv:1506.03365*, 2015.

[226] "Nonrigid ICP, MATLAB Central File Exchange, 2019.." https://www.mathworks.com/matlabcentral/fileexchange/41396-nonrigidicp/, 2019.

[227] K. Roth, A. Lucchi, S. Nowozin, and T. Hofmann, "Stabilizing training of generative adversarial networks through regularization," in *NeurIPS*, 2017.

[228] L. Mescheder, A. Geiger, and S. Nowozin, "Which Training Methods for GANs do actually Converge?," in *ICML*, 2018.

[229] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *CVPR*, 2018.

[230] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015.

[231] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," *arXiv:1710.10196*, 2017.

[232] X. Wang and A. Gupta, "Generative image modeling using style and structure adversarial networks," in *ECCV*, 2016.

[233] S. Zhu, S. Fidler, R. Urtasun, D. Lin, and C. C. Loy, "Be your own prada: Fashion synthesis with structural coherence," in *ICCV*, 2017.

[234] L. Ma, X. Jia, Q. Sun, B. Schiele, T. Tuytelaars, and L. Van Gool, "Pose guided person image generation," *arXiv preprint arXiv:1705.09368*, 2017.

[235] T. Chen, Z. Zhu, A. Shamir, S.-M. Hu, and D. Cohen-Or, "3-sweep: Extracting editable objects from a single photo," *TOG*, 2013.

[236] N. Kholgade, T. Simon, A. Efros, and Y. Sheikh, "3D object manipulation in a single photograph using stock 3D models," *TOG*, 2014.

[237] Y. Zheng, X. Chen, M.-M. Cheng, K. Zhou, S.-M. Hu, and N. J. Mitra, "Interactive images: cuboid proxies for smart image manipulation," *TOG*, 2012.

[238] H. Yu, O. G. Garrod, and P. G. Schyns, "Perception-driven facial expression synthesis," *Computers & Graphics*, 2012.

[239] R. Huang, S. Zhang, T. Li, and R. He, "Beyond face rotation: Global and local perception GAN for photorealistic and identity preserving frontal view synthesis," *arXiv preprint arXiv:1704.04086*, 2017.

[240] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *NeurIPS*, 2017.

[241] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, 2014.

[242] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," in *ECCV*, 2016.

[243] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *ICCV*, 2017.

[244] T. Kim, M. Cha, H. Kim, J. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," in *ICML*, 2017.

[245] M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures," *TOC*, 1973.

[246] P. Guan, L. Reiss, D. Hirshberg, A. Weiss, and M. Black, "Drape: Dressing any person," 2012.

[247] G. Pons-Moll, S. Pujades, S. Hu, and M. Black, "Clothcap: seamless 4d clothing capture and retargeting," in *TOG*, 2017.

[248] X. Han, Z. Wu, Z. Wu, R. Yu, and L. S. Davis, "Viton: An image-based virtual try-on network," in *CVPR*, 2018.

[249] M. Zanfir, A.-I. Popa, A. Zanfir, and C. Sminchisescu, "Human appearance transfer," in *CVPR*, 2018.

[250] N. Neverova, R. Alp Guler, and I. Kokkinos, "Dense pose transfer," in *ECCV*, 2018.

[251] R. Alp Güler, N. Neverova, and I. Kokkinos, "Densepose: Dense human pose estimation in the wild," in *CVPR*, 2018.

[252] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *SIGGRAPH*, 2000.

[253] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera, "Filling-in by joint interpolation of vector fields and gray levels," *TIP*, 2001.

[254] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, "Texture optimization for example-based synthesis," in *TOG*, 2005.

[255] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," in *SIGGRAPH*, 2001.

[256] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," *TOG*, 2009.

[257] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and Locally Consistent Image Completion," *TOG*, 2017.

[258] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," in *CVPR*, 2018.

[259] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions," *arXiv preprint arXiv:1804.07723*, 2018.

[260] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[261] N. B. Weston, "Declining sediments and rising seas: an unfortunate convergence for tidal wetlands," *Estuaries and Coasts*, 2014.

[262] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston, "Key-value memory networks for directly reading documents," *arXiv preprint arXiv:1606.03126*, 2016.

[263] T. Arici and A. Celikyilmaz, "Associative adversarial networks," *arXiv preprint arXiv:1611.06953*, 2016.

[264] C. Li, J. Zhu, and B. Zhang, "Learning to generate with memory," in *ICML*, 2016.

[265] Y. Kim, M. Kim, and G. Kim, "Memorization precedes generation: Learning unsupervised gans with memory networks," *arXiv preprint arXiv:1803.01500*, 2018.

[266] S. E. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee, "Learning what and where to draw," in *NeurIPS*, 2016.

[267] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *CVPR*, 2016.

[268] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *ECCV*, 2016.

[269] X. Mao, Q. Li, H. Xie, R. Y. Lau, and Z. Wang, "Multi-class generative adversarial networks with the L2 loss function," *arXiv preprint arXiv:1611.04076*, 2016.

[270] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[271] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *CVPR*, 2017.

[272] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, *et al.*, "Image quality assessment: from error visibility to structural similarity," *TIP*, 2004.

[273] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *NeurIPS*, 2015.

[274] O. Langner, R. Dotsch, G. Bijlstra, D. H. Wigboldus, S. T. Hawk, and A. Van Knippenberg, "Presentation and validation of the radboud faces database," *Cognition and emotion*, 2010.

[275] P. Ekman and W. Friesen, "Facial action coding system: A technique for the measurement of facial movement," *Consulting Psychologists Press*, 1978.

[276] K. R. Scherer, "Emotion as a process: Function, origin and regulation," *SSI*, 1982.

[277] S. Du, Y. Tao, and A. M. Martinez, "Compound facial expressions of emotion," *PNAS*, 2014.

[278] S. Zafeiriou, G. Trigeorgis, G. Chrysos, J. Deng, and J. Shen, "The menpo facial landmark localisation challenge: A step towards the solution," in *CVPRW*, 2017.

[279] C. F. Benitez-Quiroz, R. Srinivasan, A. M. Martinez, *et al.*, "Emotionet: An accurate, real-time algorithm for the automatic annotation of a million facial expressions in the wild.," in *CVPR*, 2016.

[280] T. Baltrušaitis, M. Mahmoud, and P. Robinson, "Cross-dataset learning and person-specific normalisation for automatic action unit detection," in *FG*, 2015.

[281] H. Zhou, Y. Liu, Z. Liu, P. Luo, and X. Wang, "Talking face generation by adversarially disentangled audio-visual representation," in *AAAI*, 2019.

[282] S. Nam, C. Ma, M. Chai, W. Brendel, N. Xu, and S. Joo Kim, "End-to-end time-lapse video synthesis from a single outdoor image," in *CVPR*, 2019.

[283] Y. Song, J. Zhu, X. Wang, and H. Qi, "Talking face generation by conditional recurrent adversarial network," *arXiv preprint arXiv:1804.04786*, 2018.

[284] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," in *NeurIPS*, 2016.

[285] K. Vougioukas, S. Petridis, and M. Pantic, "End-to-end speech-driven facial animation with temporal gans," in *BMVC*, 2018.

[286] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *ICCV*, 2015.

[287] "U.s. online shop and mail-order sales of fashion, accessories and footwear from 2003 to 2016."

[288] "Apparel, footwear and accessories sales as percentage of total retail e-commerce sales in the united states from 2016 to 2022."

[289] A. Raj, P. Sangkloy, H. Chang, J. Lu, D. Ceylan, and J. Hays, "Swapnet: Garment transfer in single view images," in *ECCV*, 2018.

[290] E. Simo-Serra, S. Fidler, F. Moreno-Noguer, and R. Urtasun, "A High Performance CRF Model for Clothes Parsing," in *ACCV*, 2014.

[291] K. Gong, X. Liang, D. Zhang, X. Shen, and L. Lin, "Look into person: Self-supervised structure-sensitive learning and a new benchmark for human parsing," in *CVPR*, 2017.

[292] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *CVPR*, 2017.

[293] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.

[294] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," *CVPR*, 2018.

[295] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," *arXiv preprint arXiv:1805.08318*, 2018.

[296] P. Esser, E. Sutter, and B. Ommer, "A variational u-net for conditional appearance and shape generation," in *CVPR*, 2018.

[297] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction," in *ECCV*.

[298] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, "Pixel2mesh: Generating 3d mesh models from single rgb images," in *ECCV*, 2018.

[299] A. Sinha, A. Unmesh, Q. Huang, and K. Ramani, "Surfnet: Generating 3d shape surfaces using deep residual networks," in *CVPR*, 2017.

[300] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "Atlasnet: A papier-mâché approach to learning 3d surface generation," *CVPR*, 2018.

[301] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling," in *NeurIPS*, 2016.

[302] H. Ben-Hamu, H. Maron, I. Kezurer, G. Avineri, and Y. Lipman, "Multi-chart generative surface modeling," in *SIGGRAPH Asia*, 2018.

[303] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image," in *CVPR*, 2017.

[304] T. Bagautdinov, C. Wu, J. Saragih, P. Fua, and Y. Sheikh, "Modeling facial geometry using compositional vaes," in *CVPR*, 2018.

[305] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *CVPR*, 2019.

[306] P. Henderson and V. Ferrari, "Learning single-image 3D reconstruction by generative modelling of shape, pose and shading," *IJCV*, 2019.

[307] P. Bojanowski, A. Joulin, D. Lopez-Paz, and A. Szlam, "Optimizing the latent space of generative networks," *PMLR*, 2017.

[308] J. Fan and J. Cheng, "Matrix completion by deep matrix factorization," *Neural Networks*, 2018.

[309] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," *ICLR*, 2017.

[310] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," in *ICASSP*, 2018.

[311] S. Kim, S.-g. Lee, J. Song, and S. Yoon, "Flowavenet: A generative flow for raw audio," *ICML*, 2018.

[312] M. Yamaguchi, Y. Koizumi, and N. Harada, "Adaflow: Domain-adaptive density estimator with application to anomaly detection and unpaired cross-domain translation," in *ICASSP*, 2019.

[313] J. Serrà, S. Pascual, and C. Segura, "Blow: a single-scale hyperconditioned flow for non-parallel raw-audio voice conversion," *NeurIPS*, 2019.

[314] H. Sun, R. Mehta, H. Zhou, Z. Huang, S. Johnson, V. Prabhakaran, and V. Singh, "Dual-glow: Conditional flow-based generative model for modality transfer," *arXiv preprint arXiv:1908.08074*, 2019.

[315] A. Grover, C. D. Chute, R. Shu, Z. Cao, and S. Ermon, "Alignflow: Cycle consistent learning from multiple domains via normalizing flows," *arXiv preprint arXiv:1905.12892*, 2019.

[316] H.-J. Chen, K.-M. Hui, S.-Y. Wang, L.-W. Tsao, H.-H. Shuai, and W.-H. Cheng, "Beautyglow: On-demand makeup transfer framework with reversible generative network," in *CVPR*, 2019.

[317] M. Kumar, M. Babaeizadeh, D. Erhan, C. Finn, S. Levine, L. Dinh, and D. Kingma, "Videoflow: A flow-based generative model for video," *arXiv preprint arXiv:1903.01434*, 2019.

[318] R. Liu, Y. Liu, X. Gong, X. Wang, and H. Li, "Conditional adversarial generative flow for controllable image synthesis," in *CVPR*, 2019.

[319] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta, "Learning a predictable and generative vector representation for objects," in *ECCV*, 2016.

[320] Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," in *CVPR*, 2019.

[321] Q. Xu, W. Wang, D. Ceylan, R. Mech, and U. Neumann, "Disn: Deep implicit surface network for high-quality single-view 3d reconstruction," in *NeurIPS*, 2019.

[322] J. Chibane, T. Alldieck, and G. Pons-Moll, "Implicit functions in feature space for 3d shape reconstruction and completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6970–6981, 2020.

[323] K. Genova, F. Cole, A. Sud, A. Sarna, and T. Funkhouser, "Local deep implicit functions for 3d shape," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4857–4866, 2020.

[324] S. Liu, S. Saito, W. Chen, and H. Li, "Learning to infer implicit surfaces without 3d supervision," in *NeurIPS*, 2019.

[325] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger, "Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision," in *CVPR*, 2020.

[326] L. Yariv, M. Atzmon, and Y. Lipman, "Universal differentiable renderer for implicit neural representations," *arXiv preprint arXiv:2003.09852*, 2020.

[327] V. Sitzmann, M. Zollhöfer, and G. Wetzstein, "Scene representation networks: Continuous 3d-structure-aware neural scene representations," in *NeurIPS*, 2019.

[328] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *arXiv preprint arXiv:2003.08934*, 2020.

[329] R. Martin-Brualla, N. Radwan, M. S. Sajjadi, J. T. Barron, A. Dosovitskiy, and D. Duckworth, "Nerf in the wild: Neural radiance fields for unconstrained photo collections," *arXiv preprint arXiv:2008.02268*, 2020.

[330] L. Liu, J. Gu, K. Z. Lin, T.-S. Chua, and C. Theobalt, "Neural sparse voxel fields," *arXiv preprint arXiv:2007.11571*, 2020.

[331] L. Yariv, Y. Kasten, D. Moran, M. Galun, M. Atzmon, B. Ronen, and Y. Lipman, "Multiview neural surface reconstruction by disentangling geometry and appearance," *NeurIPS*, 2020.

[332] M. Niemeyer, L. Mescheder, M. Oechsle, and A. Geiger, "Occupancy flow: 4d reconstruction by learning particle dynamics," in *ICCV*, 2019.

[333] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh, "Neural volumes: learning dynamic renderable volumes from images," *TOG*, vol. 38, no. 4, 2019.

[334] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017.

[335] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *arXiv preprint arXiv:1901.00596*, 2019.

[336] D. Hilbert, "Über die stetige abbildung einer linie auf ein flächenstück," in *Dritter Band: Analysis · Grundlagen der Mathematik · Physik Verschiedenes: Nebst Einer Lebensgeschichte*, 1935.

[337] N. Parmar, A. Vaswani, J. Uszkoreit, Ł. Kaiser, N. Shazeer, A. Ku, and D. Tran, "Image transformer," *arXiv preprint arXiv:1802.05751*, 2018.

[338] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[339] M. Meshry, D. B. Goldman, S. Khamis, H. Hoppe, R. Pandey, N. Snavely, and R. Martin-Brualla, "Neural rerendering in the wild," in *CVPR*, 2019.

[340] K.-A. Aliev, D. Ulyanov, and V. S. Lempitsky, "Neural point-based graphics," *arXiv preprint arXiv:1906.08240*, 2019.

[341] F. Pittaluga, S. J. Koppal, S. B. Kang, and S. N. Sinha, "Revealing scenes by inverting structure from motion reconstructions," in *CVPR*, 2019.

[342] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *CVPR*, 2016.

[343] R. Tyleček and R. Šára, "Spatial pattern templates for recognition of objects with regular structure," in *GCPR*, 2013.

[344] A. Yu and K. Grauman, "Fine-grained visual comparisons with local learning," in *CVPR*, 2014.

[345] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros, "Generative visual manipulation on the natural image manifold," in *ECCV*, 2016.

[346] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *CVPR*, 2018.

[347] K. Rematas and V. Ferrari, "Neural voxel renderer: Learning an accurate and controllable rendering tool," in *CVPR*, 2020.

[348] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[349] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *International workshop on vision algorithms*, 1999.

[350] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: exploring photo collections in 3d," in *SIGGRAPH*, 2006.

[351] M. Levoy and P. Hanrahan, "Light field rendering," in *SIGGRAPH*, 1996.

[352] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen, "Unstructured lumigraph rendering," in *SIGGRAPH*, 2001.

[353] M. Oechsle, M. Niemeyer, L. Mescheder, T. Strauss, and A. Geiger, "Learning implicit surface light fields," *arXiv preprint arXiv:2003.12406*, 2020.

[354] A. Bansal, M. Vo, Y. Sheikh, D. Ramanan, and S. Narasimhan, "4d visualization of dynamic events from unconstrained multi-view videos," in *CVPR*, 2020.

[355] S. Vedula, P. Rander, R. Collins, and T. Kanade, "Three-dimensional scene flow," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 3, pp. 475–480, 2005.

[356] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, "On the spectral bias of neural networks," in *ICML*, 2019.

[357] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *SIGGRAPH*, 1987.

[358] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *CVPR*, 2018.

[359] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter, "A 3d face model for pose and illumination invariant face recognition," in *AVSS*, 2009.

[360] E. Corona, A. Pumarola, G. Alenyà, F. Moreno-Noguer, and G. Rogez, "Ganhand: Predicting human grasp affordances in multi-object scenes," in *CVPR*, 2020.