Combining Data-Driven 2D and 3D
Human Appearance Models

Combining Data-Driven 2D and 3D Human Appearance Models

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Christoph Lassner
aus Augsburg

Tübingen
2017

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	16. März 2018
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Dr. Peter V. Gehler
2. Berichterstatter:	Prof. Dr. Hendrik Lensch
3. Berichterstatter:	Prof. Dr. Cordelia Schmid

Kurzbeschreibung

Detaillierte 2D und 3D Körperschätzung von Menschen hat vielfältige Anwendungen in unser aller Alltag: Interaktion mit Maschinen, virtuelle ‚Anprobe' von Kleidung oder direkte Produktanpassungen durch Schätzung der Körpermaße sind nur einige Beispiele. Dazu sind Methoden zur (1) detaillierten Posen- und Körpermaßschätzung und (2) Körperdarstellung notwendig. Idealerweise sollten sie digitale 2D Bilder als Ein- und Ausgabemedium verwenden, damit die einfache und allgemeine Anwendbarkeit gewährleistet bleibt. Aufgrund der hohen Komplexität des menschlichen Erscheinungsbilds und der Tiefenmehrdeutigkeit im 2D Raum sind datengetriebene Modelle ein naheliegendes Werkzeug, um solche Methoden zu entwerfen.

In dieser Arbeit betrachten wir zwei Aspekte solcher Systeme: im ersten Teil entwickeln wir allgemein anwendbare Techniken für die Optimierung und Implementierung maschineller Lernmethoden und stellen diese in Form von Softwarepaketen bereit. Im zweiten Teil präsentieren wir in mehreren Schritten, wie die detaillierte Analyse und Darstellung von Menschen basierend auf digitalen 2D Bildern bewerkstelligt werden kann.

Wir arbeiten dabei mit zwei Methoden zum maschinellen Lernen: Entscheidungswäldern und Künstlichen Neuronalen Netzen. Der Beitrag dieser Dissertation zur Theorie der Entscheidungswälder besteht in der Einführung einer verallgemeinerten Entropiefunktion, die effizient auswertbar und anpassbar ist und es ermöglicht, häufig verwendete Heuristiken besser einzuordnen. Für Entscheidungswälder und für Neuronale Netze beschreiben wir Methoden zur Implementierung und stellen jeweils ein Softwarepaket bereit, welches diese umsetzt.

Die bisherigen Methoden zur 3D Körperschätzung aus Bildern beschränken sich auf die automatische Bestimmung der 14 wichtigsten 2D Punkte, welche die Pose definieren und deren Konvertierung in ein 3D ‚Skelett'. Wir zeigen, dass durch die Optimierung einer fein abgestimmten Energiefunktion auch ein voller 3D Körper, nicht nur dessen Skelett, aus automatisch bestimmten 14 Punkten geschätzt werden kann. Damit beschreiben wir die erste vollautomatische Methode, die einen 3D Körper aus einem digitalen 2D Bild schätzt.

Die detaillierte 3D Pose, beispielsweise mit Rotationen der Körperteile und die Beschaffenheit des untersuchten Körpers, ist damit noch nicht bestimmbar. Um detailliertere Modelle zu erstellen wäre es notwendig, Daten mit einem hohen Detailgrad zu annotieren. Dies skaliert jedoch nicht zu großen Datenmengen, da sowohl der Zeitaufwand pro Bild, als auch die notwendige Qualität aufgrund der schwierig einzuschätzenden exakten Positionen von Punkten auf der Körperoberfläche nicht erreicht werden können. Um dieses Problem zu lösen entwickeln wir eine Methode, die zwischen der Optimierung der 2D und 3D Modelle alterniert und diese wechselseitig verbessert. Dabei bleibt der Annotationsaufwand für Menschen gering. Gleichzeitig gelingt es, 2D Modelle mit einem Vielfachen an Details bisheriger Methoden zu erstellen und die Schätzung der 3D Pose und des Körpers auf Rotationen und Körperumfang zu erweitern.

Um Bilder von Menschen zu generieren, beschränken sich existierende Methoden auf 3D Modelle, die schwer anzupassen und zu verwenden sind. Im Gegensatz dazu nutzen wir in dieser Arbeit einen Ansatz, der auf den Möglichkeiten zur automatischen 3D Posenschätzung basiert: wir nutzen sie, um einen Datensatz aus 3D Körpern mit dazugehörigen 2D Kleidungen und Kleidungssegmenten zu erstellen. Dies erlaubt es uns, ein datengetriebenes Modell zu entwickeln, welches direkt 2D Bilder von Menschen erzeugt.

Erst das vielfältige Zusammenspiel von 2D und 3D Körper- und Erscheinungsmodellen in verschiedenen Formen ermöglicht es uns, einen hohen Detailgrad sowohl bei der Analyse als auch der Generierung menschlicher Erscheinung zu erzielen. Die hierfür entwickelten Techniken sind prinzipiell auch für die Analyse und Generierung von Bildern anderer Lebewesen und Objekte anwendbar.

Abstract

Detailed 2D and 3D body estimation of humans has many applications in our everyday life: interaction with machines, virtual try-on of fashion or product adjustments based on a body size estimate are just some examples. Two key components of such systems are: (1) detailed pose and shape estimation and (2) generation of images. Ideally, they should use 2D images as input signal so that they can be applied easily and on arbitrary digital images. Due to the high complexity of human appearance and the depth ambiguities in 2D space, data driven models are the tool at hand to design such methods.

In this work, we consider two aspects of such systems: in the first part, we propose general optimization and implementation techniques for machine learning models and make them available in the form of software packages. In the second part, we present in multiple steps, how the detailed analysis and generation of human appearance based on digital 2D images can be realized.

We work with two machine learning methods: Decision Forests and Artificial Neural Networks. The contribution of this thesis to the theory of Decision Forests consists of the introduction of a generalized entropy function that is efficient to evaluate and tunable to specific tasks and allows us to establish relations to frequently used heuristics. For both, Decision Forests and Neural Networks, we present methods for implementation and a software package.

Existing methods for 3D body estimation from images usually estimate the 14 most important, pose defining points in 2D and convert them to a 3D 'skeleton'. In this work we show that a carefully crafted energy function is sufficient to recover a full 3D body shape automatically from the keypoints. In this way, we devise the first fully automatic method estimating 3D body pose and shape from a 2D image.

While this method successfully recovers a coarse 3D pose and shape, it is still a challenge to recover details such as body part rotations. However, for more detailed models, it would be necessary to annotate data with a very rich set of cues. This approach does not scale to large datasets, since the effort per image as well as the required quality could not be reached due to how hard it is to estimate the position of keypoints on the body surface. To solve this problem, we develop a method that can alternate between optimizing the 2D and 3D models, improving them iteratively. The labeling effort for humans remains low. At the same time, we create 2D models reasoning about factors more items than existing methods and we extend the 3D pose and body shape estimation to rotation and body extent.

To generate images of people, existing methods usually work with 3D models that are hard to adjust and to use. In contrast, we develop a method that builds on the possibilities for automatic 3D body estimation: we use it to create a dataset of 3D bodies together with 2D clothes and cloth segments. With this information, we develop a data driven model directly producing 2D images of people.

Only the broad interplay of 2D and 3D body and appearance models in different forms makes it possible to achieve a high level of detail for analysis and generation of human appearance. The developed techniques can in principle also be used for the analysis and generation of images of other creatures and objects.

Danksagungen

Diese Arbeit ist zum Teil in den ersten drei Jahren meiner Doktorandenzeit an der Universität Augsburg und zum Teil in den letzten zwei Jahren am Bernstein Center for Computational Neuroscience und am Max-Planck Institut für Intelligente Systeme in Tübingen entstanden.

Mein erster Dank gilt meinem Betreuer in Tübingen, Dr. Peter Gehler, für seine vielfältige Förderung und Unterstützung und seine einzigartige Inspiration, die er mit mir geteilt hat. Gleichermaßen bedanke ich mich bei meinem Betreuer an der Universität Augsburg, Prof. Rainer Lienhart, für seine Förderung und Unterstützung. Ich erinnere mich gerne an die Zeit an beiden Institutionen; weder diese Arbeit, noch die darin vorgestellten Systeme wären ohne die Fähigkeiten, die ich bei beiden Gruppen aus Augsburg und Tübingen erlernen durfte, entstanden. Mein besonderer Dank in dieser Hinsicht gilt Prof. Michael Black, der es mir durch seine Förderung und Zusammenarbeit mit seiner "Body Group" ermöglicht hat, den Bogen zwischen 2D und 3D Körpermodellen zu schlagen, was zum essentiellen Bestandteil dieser Arbeit wurde. Mein weiterer Dank gilt Prof. Hendrik Lensch, Prof. Cordelia Schmid, Prof. Matthias Bethge und Prof. Andreas Schilling für ihre Unterstützung und ihr Feedback.

Bei meinen Kollegen in Augsburg und Tübingen möchte ich mich für die tollen wissenschaftlichen Diskussionen, aber auch die gute Atmosphäre und die entstandenen Freundschaften bedanken. Ich erinnere mich gerne an die gemeinsamen Stunden im Unikum in Augsburg mit Stefan Romberg, Fabian Richter, Dan Zecha und Christian Eggert. Dan's "weniger Abhängigkeiten sind mehr" wird mich weiterhin begleiten. In Tübingen danke ich besonders Martin Kiefel, Fatma Güney, Thomas Nestmeyer, Laura Sevilla-Lara, Varun Jampani und Sergey Prokudin für die gemeinsam verbrachte schöne Zeit. Darüber hinaus möchte ich mich bei Melanie Feldhofer für ihre organisatorische Unterstützung und zielstrebige Gelassenheit in allen Situationen bedanken; und natürlich genauso bei unserem Rocko "the rock". Bei der Association for Computing Machinery (ACM) bedanke ich mich für die finanzielle Unterstützung in Form eines Student Travel Grants.

Ein besonderer Dank gilt meinen Förderern (chronologisch): Dr. Roland Kircher und Dr. Claus Bahlmann von der Siemens AG, die mich für Computer Vision und Modeling begeistert haben, Dr. Sebastian Nowozin für immer wieder tolle Denkanstöße und Prof. Bernt Schiele für seine Förderung, tolle Retreats und Boulder-Sessions.

Zum Schluss möchte ich mich bei meiner Familie und Freunden bedanken für die tolle Unterstützung in allen Lebenslagen. Ganz besonders bei meinen Eltern für ihre Weitsicht und Offenheit, die sie mit mir geteilt und die Möglichkeiten, die sie mir dadurch geschaffen haben.

Tübingen, 2018 *Christoph Lassner*

Contents

Notation and Symbols

The overall notation and symbols are introduced here. For a brief method-specific introduction, see Chapter 2.

General Notation

Scalars	Regular (greek) lower case, *e.g.*, a, b, c, δ, η.
Vectors	Bold (greek) lower case, *e.g.*, $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{\delta}, \boldsymbol{\eta}$.
Vector element	Subscript index, *e.g.*, a_i: the i-th element of $\boldsymbol{a}$.
$\mathbf{0}$	Zero vector.
Matrices	Bold (greek) upper case, *e.g.*, $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{\Lambda}, \boldsymbol{\Sigma}$.
Matrix element	Row, column indices in subscript, *e.g.*, $\boldsymbol{A}_{i,j}$. If only one subscript is provided, it refers to the i-th element in flat row-major indexing.
$\boldsymbol{I}$	Identity matrix.
Sets	Calligraphic upper case, *e.g.*, $\mathcal{A}, \mathcal{B}, \mathcal{C}$.
Dimensionality, Cardinality	Regular upper case, *e.g.*, D, M.

Numbers

$\mathbb{N}$	Natural numbers, $0 \notin \mathbb{N}$.
$\mathbb{Z}$	Integers.
$\mathbb{R}$	Real numbers.

Probabilities

θ	Model parameters.
θ_*	Most probable A Posteriori (MAP) setting for θ.
$\mathcal{D}$	Data, evidence.
$\mathfrak{p}(\cdot)$	Probability density function.
$\mathcal{N}$	Gaussian distribution.
$\mathcal{U}$	Uniform distribution.
$x \sim \mathcal{T}$	Random variable x is distributed according to $\mathcal{T}$.
$\boldsymbol{E}[\cdot]$	Expected value.
$\mathbf{var}[\cdot]$	Variance.
$\mathbf{cov}[\cdot]$	Covariance.

Decision Forests

w	Number of classes.
$h(\boldsymbol{v}, \theta)$	Decision function;
	arg.: data vector $\boldsymbol{v}$ and decision parameters θ.
$H(\boldsymbol{p})$	Discrete entropy function;
	arg.: vector of class probabilities $\boldsymbol{p}$.
$H[\mathfrak{p}]$	Differential entropy function;
	arg.: probability density function $\mathfrak{p}$.
$S(\boldsymbol{p})$, $S[\mathfrak{p}]$	Shannon entropy.
$R_q(\boldsymbol{p})$, $R_q[\mathfrak{p}]$	Rényi entropoy.
$T_q(\boldsymbol{p})$, $T_q[\mathfrak{p}]$	Tsallis entropy.
$N_q(\boldsymbol{p})$, $N_q[\mathfrak{p}]$	Norm-induced entropy.
$C(\boldsymbol{p})$	Classification error.

Part I

Introduction

Chapter 1

Motivation

As one of the most versatile tools, computers take an increasingly important role in our society. They manage our data, enable product design and development and are helping to connect and entertain people. With improvements in computing power per volume and with improving energy efficiency, we already speak of 'ubiquitous computing', reflecting the presence of computers in many areas of everyday life. This trend can be expected to continue and accelerate with progress in the areas of artificial intelligence and robotics. With the current race towards autonomous driving[1] and plans to support the elderly with automatic systems[2], these topics impact the whole society.

At the same time, the need for more complex and safe interaction patterns rises. If computers should 'participate' in everyday life, a substantial understanding of human behavior and intent becomes imperative to ensure a convenient and pleasant interaction and environment: an autonomous car must understand where a human wants to go or move, where a bicyclist wants to go next or if kids are involved in a game on the roadside that may cause them to jump on the street in the next, critical seconds. This is just an enumeration of example scenarios that can be extended by countless situations in work or private environments.

Everyday interaction patterns must remain natural, *i.e.*, they must work without additional cues from specific hardware that is worn on the body or operated in a specific way. If interaction can possibly happen at all times and security is a factor, there may be no specific requirements to make it work. Hence, visual information is a natural and sufficient choice to base this interaction on. Sole human detection or reasoning about the 2D pose in an image is not sufficient for this task: the 3D pose and ideally details about the whole body must be parsed to develop a detailed understanding of a human.

Vice versa, many applications may benefit from a realistically rendered virtual person. This ranges from plain interaction to creating summaries and animations of your own body development over time or a virtual fitting room.

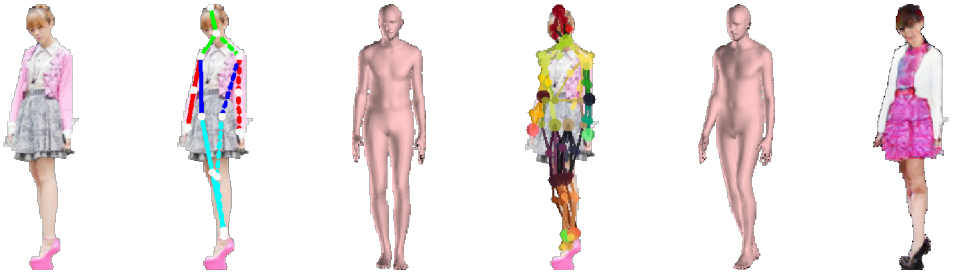
[1] `http://wapo.st/2orBvAW?tid=ss_mail&utm_term=.27ff0682d90b`
[2] `http://newsroom.toyota.co.jp/en/detail/8709541/`

With this thesis, we build on the latest developments in artificial intelligence to propose novel techniques to reason about 2D and 3D human body pose, shape and appearance in general. We develop new building blocks for perception and interaction in the most natural way: with our body. In the long run, the presented technology will enable machines to parse fine nuances of human behavior to better understand us and provide us with easier ways to communicate.

1.1 Problem Statement

We want to enable automatic computing systems to link visual evidence in 2D and 3D to an informative, low-dimensional representation of human appearance and vice versa. As examples, the developed techniques may answer or solve the following questions and tasks given an image:

- Where are people in this image?
- In which (3D) direction is this person pointing?
- How far away from the camera is this person?
- Is this person sitting?
- What kind of clothes is this person wearing?
- Generate ten people with different clothing in the same pose and with the same body shape as this person.



(a) **(b)** **(c)** **(d)** **(e)** **(f)**

Figure 1.1: *Illustrative example for detailed person appearance analysis and synthesis.* **(a)** original image, **(b)** 14 keypoint detections [IPA+16], **(c)** automatic 3D body model fit (Chapter 6). Due to the sparse keypoints and the not invertible perspective projection, the orientation of the person cannot be resolved. **(d)** 91 keypoint detections and **(e)** improved fit (Chapter 8), **(f)** automatically generated redressed person with same body pose and shape (Chapter 9). The 3D body model in (c) carries more information than the 14 keypoints in (b). Using this as a foundation, we improve the level of detail to refine the internal human representation (d) and the 3D body model fit (e). The resulting datasets and techniques help us to build an entirely data-driven 2D generative model, conditioned on 3D data (f).

As main modality, we work with digital 2D images, since 2D capture devices are ubiquitous and data is easy to acquire. Furthermore, the developed technologies are not depending on possibly expensive hardware.

Parts of this task have been extensively studied in isolation over the last years. There is substantial literature on identifying the 14 keypoints defining the main skeleton joints of a person [FH05, ARS09, JE10, PJA+12, YR13, DGL14, APGS14, WRKS16, IPA+16, NYD16] (Figure 1.1b). These 14 2D keypoints help to localize people and body parts in 2D. They are insufficient to reason about detailed 3D body pose or shape, because they do not resolve perspective ambiguities and they lack information about body extent (*e.g.*, Figure 1.1c). Several works on lifting 2D keypoints to 3D pose solely worked on the 14 skeleton points and do not take body shape into account [RKS12, AB15, ZZLD15] or require additional shape information [SBB08, GWBB09, Gua12, HAR+10, ZFL+10, CKC10] and often manual hints. In contrast, our methods can use silhouette cues if available and is fully automatic. Semantic image segmentation, *i.e.*, segmenting areas with people in an image or identifying areas with body parts or clothes, has been studied usually independent of keypoints [EGW+10, YKOB12, CML+14, SSFMNU14, CPK+15, LXS+15, OVB+16]. Garment and cloth modeling has been mostly considered in 3D [dASTH10, SGdA+10, WOR11, GRH+12, KKN+13].

Few existing works aim to bridge the gaps between these fields and usually do not generalize to everyday scenarios (for more complete reviews of existing literature, please refer to the related work sections of the respective chapters). In contrast to these existing systems, we aim to (1) build robust systems that work in everyday, non-laboratory environments and (2) work with a more complete 3D representation than skeletons. Illustrative example results for selected methods developed in this thesis can be found in Figure 1.1.

1.2 Applications

We briefly present three groups of particularly interesting application areas:

Human Machine Interaction (HMI) Any application with 3D human machine interaction possibly benefits from the technologies presented in this thesis. They enable a more detailed understanding of human pose and shape, which may be of interest for safety or convenience reasons. In terms of safety, self-driving cars are an example of currently particularly high interest. Another area is shared workplaces for robots and humans and support for and interaction with physically disabled people. One of the major manufacturer for industrial robots participated in the opening ceremony for the Paralympics 2016[3] to emphasize their interest in intensifying their work in this area.

[3] https://www.kuka.com/en-de/press/news/2016/09/kuka-robot-dances-at-2016-paralympic-opening-ceremony

Virtual realities and telepresence Virtual reality depends on an immersive experience. Hardware attached to the body counters natural perception of the artificial environment. Instead, detailed 3D pose and body shape estimation can provide the information to render the own 3D body in the virtual environment. This leads to a more realistic self perception and also extends to the perception of other people in the virtual environment. Their bodies can be perceived and rendered in the virtual world with similar techniques[4].

Customization and fitting The presented techniques can be used to try on clothes and other products virtually. This could be implemented either by using estimated 3D body shape and pose or by directly parsing and altering a 2D camera image with new clothes. Additionally, products or virtual characters may be customized based on 3D body shape or pose.

Some of these applications have rigorous demands for short processing times which are not met yet. However, we have no reason to believe that further optimization could not accelerate the presented techniques to sufficient speeds. In Section 8.3.5 and Chapter 11, we discuss promising ideas in this direction. The rapid development of potent hardware specifically for AI applications, even for mobile platforms, is further evidence suggesting that low latency solutions may be available soon[5,6].

1.3 Challenges

All parts of the aforementioned task are natural for humans to solve and we do so many times on a daily basis. To solve them, we build on a lot of experience and training and on stereoscopic information from our eyes. In the following paragraphs, we want to briefly provide an intuition on the challenges needed to overcome to build systems solving these problems in an automatic manner from digital 2D inputs.

In general, the image formation process itself is not explicitly invertible due to the perspective projection: depth information is irretrievably lost. Additionally, digital images are usually represented in a uniformly spaced, three-channel 8-bit (255 value) format (see Figure 1.2c). Reconstructing 3D body shape information from these values without semantic meaning is a challenging task and makes explicitly coded solutions very hard or impossible to find. Instead, we use data-driven solutions based on machine learning.

Even with these expressive methods, human appearance is inherently difficult to model due the diversity in clothing and poses. Clothing is particularly challenging

[4]Microsoft experimented with similar setups already: `https://blogs.msdn.microsoft.com/kinectforwindows/2016/03/07/kinect-lets-you-see-yourself-in-vr-game/`. This approach depends on the Kinect sensor and is restricted to indoor environments.

[5]`https://cloud.google.com/tpu/`

[6]`https://www.bloomberg.com/news/articles/2017-05-26/apple-said-to-plan-dedicated-chip-to-power-ai-on-devices`

(a) T-Shirt printed to emulate depth [The16a]. Strong patterns or depth emulation can hide true depth structures.

(b) Yoga pose [Mil05]. Human appearance strongly varies with pose.

(c) Representation as pixel values carries no semantic information and generalizes far beyond natural images.

Figure 1.2: *Challenges for the automatic perception of humans.*

to parse as well as to generate, with extreme cases of, *e.g.*, emulated 3D pictures on T-Shirts (an example can be found in Figure 1.2a). They are impossible to parse without context. Hence, we build models that take context into account, namely Convolutional Neural Networks (CNNs). The pose diversity is a similarly challenging problem. Extreme examples can be found, *e.g.*, in Yoga (for an example, see Figure 1.2b). Whereas the pose in this example is certainly not an every day pose to assume, it illustrates how much pose can change the appearance of a human. In non-sport scenarios, poses are usually strongly limited and we can use priors to distinguish between likely and unlikely poses.

To make data-driven solutions work, large datasets are required. Even though there are plenty of digital images of humans, acquiring labeled data is not trivial, especially in 3D. Hence, we suggest the following split throughout the presented methods in this thesis: first, move from image space to a semantic representation in 2D, then 'lift' the data on the semantic level to 3D. This elegantly sidesteps capturing realistic 3D data including labels; instead, we generate data in the semantic space. With this, we present a blueprint on how to fuse data-driven 2D and 3D human appearance models in an elegant way.

1.4 Organization and Contributions

The contributions of this thesis lie in between the fields of machine learning and human appearance modeling. This fact is reflected in the structure which is split into two parts: Part II describes contributions towards machine learning techniques and machine learning frameworks whereas Part III focuses on 2D and 3D human appearance models.

The contributions for learning techniques are the following:

- We show that two of the four Khinchin-Shannon axioms are sufficient to model the requirements for an entropy function for Decision Forest training. This leads to a well-defined family of generalized entropies: the norm-induced entropies. We introduce the *norm-induced entropy* and analyze its properties, show relations to existing entropies and heuristics and analyze its performance in Decision Forest training.
- We propose an object oriented implementation of Decision Forests. This could be especially helpful for applications in research where on the one hand well-tested components can be reused and on the other hand it is easy to modify selected parts to implement and evaluate new ideas. Code is available at [LL14].
- The *caffe* deep learning framework [JSD+14] is widely used in the computer vision community. It heavily relies on several *protobuf* model files that are hard to maintain and to keep consistent. We propose the *barrista* framework that allows to design and use models fully in the Python programming language. Code is available at [LKKG15].

For combining 2D and 3D human appearance models, we claim the following contributions:

- We present the first fully automatic method of estimating 3D body shape and pose from 2D joints. For this purpose, we develop an energy function matching a 3D body model to 2D joints. It includes a differentiable interpenetration term and optionally a term that takes foreground-background information into account. Code is available at [BKL+16b].
- The fitting process solely to 2D joints can not fully reason about detailed body shape. Segmentation information can provide the necessary cues. We present a new human semantic (part) segmentation dataset for diverse poses and appearances consisting of 19,652 image annotations built on datasets with existing keypoint annotations. Furthermore, we present experiments on fusing joint and segmentation information and present a method to incorporate segmentation cues in the 3D fitting method.
- For fine-grained 3D body pose and shape analysis, detailed annotations are necessary. We propose to use an iterative alternation between training a discriminative model, using it to automatically fit 3D bodies on large scale, curate these by human annotators, projecting the 3D data to 2D and iterate. In our experiments, this strategy allowed us to step factors beyond the number of segments/keypoints in existing work with reliable models. Furthermore, the high number of keypoints allowed us to develop an at test-time optimization free method for predicting 3D body pose and shape with reasonable prediction performance. This results in a great speed-up for 3D body pose and shape prediction from 2D images. Code and data are available at [LRK+16].

- We develop the first image-based generative model of people in clothing in a full-body setting. It allows to either sample people unconditioned or conditioned on 3D body pose and shape. Code and data available at [LPMG17b].

1.5 List of Publications

Parts of this thesis are based on earlier works as listed below. Additional results and figures have been added and the text has been partially adjusted for a better presentation in the context of this work.

- C. Lassner, R. Lienhart. Norm-induced Entropies for Decision Forests. WACV 2015 [LL15a] © 2015 IEEE `http://dx.doi.org/10.1109/WACV.2015.134`. Presented in Chapter 3.
- C. Lassner, R. Lienhart. The fertilized forests Decision Forest Library. ACM Multimedia Open-Source Software Competition 2015 [LL15b] © 2015 ACM `http://dx.doi.org/10.1145/2733373.2807407`. *Honorable Mention.* Presented in Chapter 4.
- C. Lassner, D. Kappler, M. Kiefel, P. V. Gehler. Barrista – Caffe well-served. ACM Multimedia Open-Source Software Competition 2016 [LKKG16] © 2016 ACM `http://dx.doi.org/10.1145/2964284.2973803`. Presented in Chapter 5.
- F. Bogo, A. Kanazawa, C. Lassner, J. Romero, P. V. Gehler, M. J. Black. Keep it SMPL: Automatic Estimation of 3D Human Pose and Shape from a Single Image. ECCV 2016 [BKL+16a]. Summarized in Chapter 6.
- C. Lassner, J. Romero, M. Kiefel, F. Bogo, M. J. Black, P. V. Gehler. Unite the People – Closing the Loop Between 3D and 2D Human Representations. CVPR 2017 [LRK+17] © 2017 IEEE `http://dx.doi.org/10.1109/CVPR.2017.500`. Presented in Chapter 8 and Section 7.5.
- C. Lassner, G. Pons-Moll, P. V. Gehler. A Generative Model of People in Clothing. ICCV 2017 [LPMG17a] © 2017 IEEE `http://dx.doi.org/10.1109/ICCV.2017.98`. Presented in Chapter 9.

I (co-)authored the following publications during my time as PhD student which are not covered by this thesis:

- C. Lassner, R. Lienhart. Methods and Applications for Distance Based ANN Training. ICMLA 2013 [LL13].
- A. Schiendorfer, C. Lassner, G. Anders, W. Reif, R. Lienhart. Active Learning for Abstract Models of Collectives. SAOS 2015 Workshop [SLA+15a].
- A. Schiendorfer, C. Lassner, G. Anders, W. Reif, R. Lienhart. Active Learning for Efficient Sampling of Control Models of Collectives. SASO 2015 [SLA+15b].

- M. Mahsereci, L. Balles, C. Lassner, P. Hennig. Early Stopping Without a Validation Set. ArXiv 2017 [MBLH17].
- Y. Huang, F. Bogo, C. Lassner, A. Kanazawa, P. V. Gehler, J. Romero, I. Akhter, M. J. Black. Towards Accurate Marker-less Human Shape and Pose Estimation over Time. 3DV 2017 [HBL+17].

Chapter 2

Techniques and Notation

In this chapter, we briefly introduce notation and basic techniques. This discussion has no claim on completeness, but is merely an overview with highlights on important or historic aspects of the models and pointers to literature.

2.1 Decision Forests

Random Forests have been developed by Leo Breiman [Bre01] in 2001. Since then, they are a frequently used statistical tool valued for its robustness and versatility. In this thesis, we will use the term Decision Forests instead of Random Forests, which is a notion introduced by A. Criminisi and J. Shotton [CS13]. We use this name to emphasize the closeness of our notation to their newer but equivalent formulation.

Decision Forests are averaging ensembles of Decision Trees [BFSO84]. A Decision Tree is a (w.l.o.g.) binary tree-structured graph. Data points $\boldsymbol{v}$ are forwarded through the graph from the root node to a leaf. At each inner node, its path is determined by a binary decision function $h(\boldsymbol{v}, \theta) : \mathbb{R}^D \times \mathcal{T} \to \{\text{left}, \text{right}\}$, usually a simple threshold with parameters $\theta \in \mathcal{T}$ on one of the D data features (for other choices and a more complete discussion, we refer to [CS13]). At a leaf node, a model for the labels of all samples from the training set having arrived at the node is stored. For the example tree given in Figure 2.1, the sample $(0, 1)$ would be associated with the leftmost leaf and the sample $(1, 1)$ with the rightmost leaf (notation: (v_0, v_1), a tuple of feature 0 and feature 1 of data point $\boldsymbol{v}$).

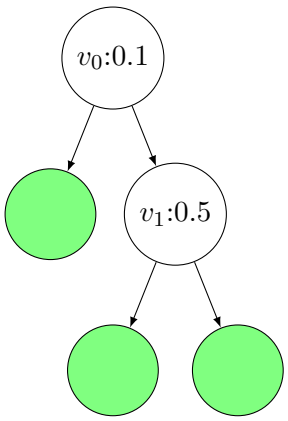
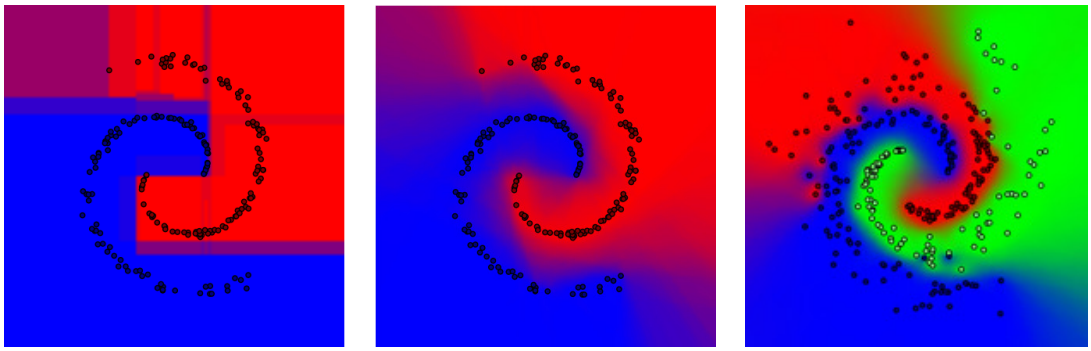


Figure 2.1: *Illustrative example Decision Tree.* **White:** inner node; **green:** leaf node. For the inner nodes, the labels specify 'feature:threshold'.



(a) **(b)** **(c)**

Figure 2.2: *The positive effect of ensemble averaging and complex features for Decision Forests.* **(a)** Result of 10 trees with axis aligned decision boundaries. Edges can easily be spotted. **(b)** Result of 200 trees with linear decision boundaries. **(c)** Result of 200 trees with quadratic decision boundaries in a three class setting.

Usually, a greedy training strategy is used to create the trees, which does not allow to optimize a global loss function (with the few notable exceptions of [SL99, Fri01, SWL+13, KFCR15]). An entropy measure $H(\boldsymbol{p})$ or $H[\mathfrak{p}]$ is used to evaluate the impurity of labels for a set of data points, where $\boldsymbol{p}$ denotes the class probabilities in a classification and $\mathfrak{p}$ the probability distribution of values in a regression setting. At an inner node, several features and thresholds are evaluated. The threshold value and feature with the lowest weighted linear combination of entropy values in the subsets is used as split criterion. If less than a predefined number of samples are routed to a node, a leaf node with a model for these samples is created. You can find visualizations of example prediction results for the 2D spiral toy problem in Figure 2.2.

Decision Forest models have two main advantages[1]: (1) training and evaluation is fast, because of the divide-and-conquer principle. (2) Domain and image of the function to approximate can be arbitrarily complex: as long as local approximations by the leaf models are valid, the performance of the model will be good. This is different from other popular machine learning methods, *e.g.*, Artificial Neural Networks (ANNs), that suffer more from such issues and are more susceptible to numerical instabilities.

These advantages, however, come at a price. Due to the divide and conquer strategy, the subsets of data routed to decision nodes at lower levels of the tree are quickly reduced in size. This means that it is a lot harder to find representative features at these stages. Also, a single noisy feature value for a decision in early levels in a tree may irrevocably falsify the predicted result. This effect is dampened by the forest ensemble.

[1]In exploratory data analysis, the additional advantage of compact, interpretable models is important. However, due to a lack of semantic meaning of the features, this advantage is hardly relevant in computer vision applications.

The weaknesses are particularly disadvantageous for tasks with image inputs since the forests do not learn to extract complex features, in contrast to CNNs. Nevertheless, for certain problems Decision Forests remain the model of choice. We use them in Section 7.4.2 and Section 8.3.5 for tasks where feature learning is not required. For these tasks, we noted superior performance in comparison with ANNs. Additionally, probabilistic leaf models allow to get confidence intervals for predictions—a feature that is not easy to implement for neural networks.

2.2 Deep Convolutional Neural Networks

Deep convolutional neural networks are the current de facto standard for learning functions with 2D image inputs or outputs. A. Krizhevsky, I. Sutskever and G. Hinton started a streak of success for these models with their results on the ImageNet challenge [DDS+09] in 2012, where they outperformed other state-of-the art models with a 36% lower error rate [KSH12]. Neural networks are based on the idea to combine many *perceptrons* [Ros58]. The concept is illustrated in Figure 2.3.

For every differentiable loss function, Stochastic Gradient Descent (SGD) or SGD-based optimizers such as ADAM [KB14] can be used to optimize $\boldsymbol{w}$. Initially, saturating but fully differentiable functions have been used for the nonlinearity, *e.g.*, the sigmoid or tanh functions. One of the important contributions of [KSH12] was that a Rectified Linear Unit (ReLU), $\text{relu}(x) = \max(x, 0)$ could be used as nonlinearity with better generalization behavior and fast evaluation even though the gradient

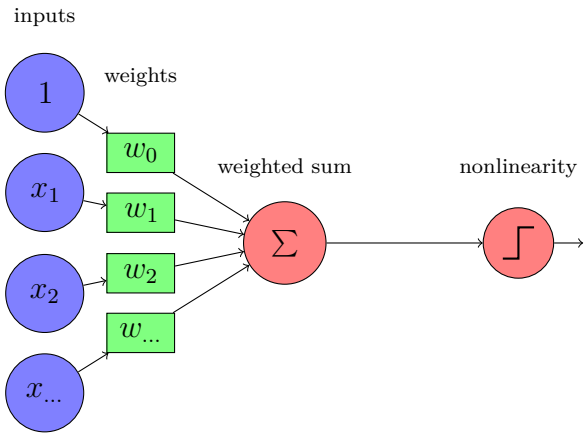


Figure 2.3: *Example perceptron.* The first processing step is the calculation of a weighted sum of inputs (**blue**) with adjustable weights (**green**). The inputs include a constant 1 value, effectively resulting in an adjustable bias. Finally, a nonlinearity is applied. Perceptrons can be stacked and then become universal approximators for bounded, continuous functions [Cyb89, Hor91].

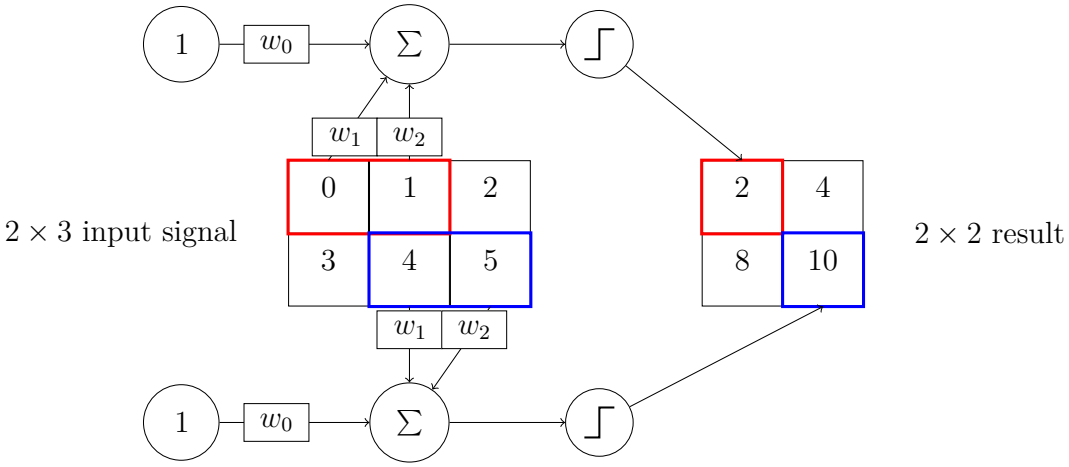


Figure 2.4: *Visualization of a* 1×2 CNN *convolution operation.* **Red** highlights inputs and outputs for the first, **blue** for the last position of the mask. It is moved consecutively over all positions on the input signal. The weights at each point of application are shared. On the right hand side example outputs are given assuming that all weights are equal to one.

is zero in large parts of its Domain. We use the Leaky ReLU (LReLU) [MHN13]

$$\text{lrelu}(x; d) = \begin{cases} x & \text{if } x > 0 \\ d \cdot x & \text{otherwise} \end{cases} \tag{2.1}$$

with dampening factor $d = 0.2$ in Chapter 9 to avoid the zero gradient problem.

Fukushima [Fuk80] and LeCun [LBBH98] introduced and popularized the convolutional application of perceptrons (an illustrative example is given in Figure 2.4). In this scenario, a perceptron has connections to a fixed area of the full input signal (in the example a 2×1 area). This area is then shifted over the full signal. The perceptron is used at every position with the same shared weights and the results are stored in an appropriately sized output matrix. During forward propagation, this corresponds to a convolution operation with the mirrored weight matrix of the perceptron and applying the nonlinearity on the result. This has several advantages: compared to a perceptron connected to the full input signal, the number of parameters is strongly reduced (in the minimal example in Figure 2.4 from 6 weights for a fully connected perceptron to 2 weights using the convolution). At the same time, the same weights are applied at many input positions, resulting in a virtually higher number of training examples (virtually, because during the back propagation phase 'real' training examples and 'convolution positions' are treated differently and additional effects occur for hidden layers). Additionally, the convolutions capture the continuous characteristics of images better than fully connected perceptrons.

The move to deep learning Usually, several of the convolutional filters are used on the same input but with different initializations: they form a 'layer'. Over the course of the optimization, they converge to different filters with different responsibilities for the task at hand. Their outputs are stacked and can be used as input to a following layer. The output of each convolution forms a 'channel' of the result. Currently, up to 1000 layers can be processed and trained on a GPU, but usually networks with around 150 layers are used [HZRS16a].

To create state-of-the art models, a few more building blocks are used:

Dropout [KSH12] A fixed percentage of layer results (*e.g.*, 50%) are zeroed out randomly per sample. Consequently, they do not contribute to the gradients if they are 'dropped'. The network must learn to be robust against these missing values, hence is less likely to converge to instable local minima. This is a drastic but successful strategy to combat overfitting. It is usually applied only during training and only on the last layers of a network.

Strided convolution The intuition for this technique as well as pooling (see below) is to reduce the resolution of the result and enable following filters to cover a higher percentage of the input signal. The strided convolution is performed by moving the perceptron mask with 'gaps' over the input.

Pooling Pooling is applied to a fully convolved result and uses a summary statistic on a local area, usually a max operation applied on a 2×2 area. This results in a reduction by factor 2 in width and height. The gradient is sparsely backpropagated solely to the maximum element. To counter this effect, average or sum pooling can be used.

Batch normalization [IS15] This technique has been introduced by S. Ioffe and C. Szegedy. The idea is to normalize the output of each layer channel-wise w.r.t. mean and variance across the processed batch. This has several advantages, but most importantly, it (1) makes the input to the following layer more independent of the current state of the preceding ones and (2) it counters the effect of vanishing gradients through deep architectures. Using batch normalization, often a higher learning rate can be used during training, resulting in faster convergence. For further information, we refer to [GBC16].

Deconvolution/fractionally strided convolution [LSD15] The step wise reduction of resolution is an important technique to enable a model to reason about long-distance relations. For tasks that require results in a similar size as inputs, the reduction must be inverted to produce a result in sufficient resolution. Here, the fractionally strided convolution can be used. The idea is to use a 'fractional' stride of $\frac{1}{x}$, $x \in \mathbb{N}$ that can be simulated by having an 'inner padding' within the image grid. This results in a size increase by factor x.

We use combinations of these techniques as required throughout Part III to build deep neural networks.

CNNs deliver the best performance for predictions from image space for most tasks to date. At the same time, parts of their behavior are not fully understood. An example for this are adversarial examples [SZS+14, MDFFF17]. Similarly, the design space for deep neural networks is extremely large. Structures are discovered that produce high quality results with a fraction of parameters and computation effort of large models (*e.g.*, [IMA+16]). It is even disputed whether deeper (more stacked layers) or wider (more filters) networks have superior performance [ZK16]. Wide networks build on the hypothesis that a lot of complex interactions manifest directly in detectable patterns in the plain image. They are advantageous from a computational perspective, since many operations in these networks can be performed in parallel. Deep networks have consecutive dependencies on previous layer results, requiring serial execution.

2.3 3D Body Models

To reason about 3D body shape and pose, we need a human body model that captures these parameters well. Commercial models from the game industry are usually not well suited for this task, because they have a different focus: realistic or stylized clothing and recognizable characters. Anatomic correctness is secondary in these cases, which manifests in 'stiff' looking animations or unrealistic tissue and joint deformations in certain poses. For our applications, we need a model that is carefully crafted to capture natural body shape including pose dependent deformations.

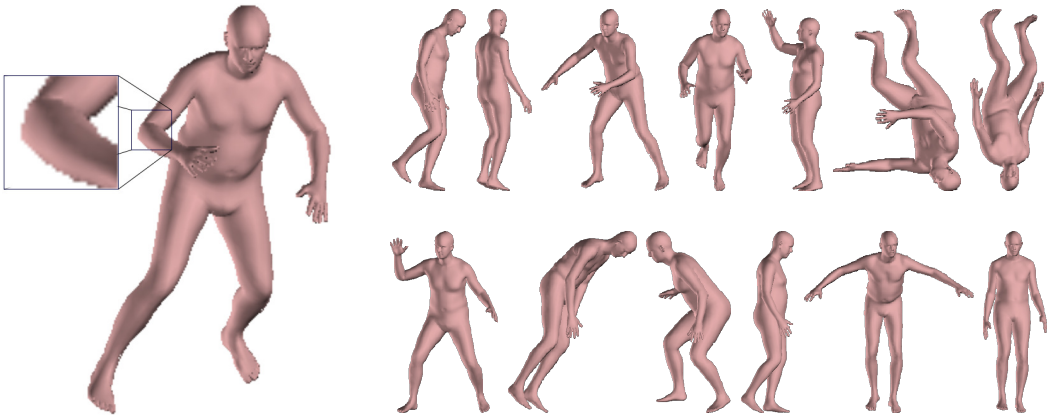


Figure 2.5: *The SMPL 3D human body model [LMR+15].* **Left:** blend shapes are used to produce realistic results for a wide variety of poses, *e.g.*, for the bent elbow. **Right:** example body fits in a variety of poses and with different shapes from the UP-3D dataset introduced in Chapter 8.

One of the first such models was SCAPE [ASK+05]. It was built from a series of 3D scans and could model muscle deformations. A series of works [HAR+10, CLZ13, PMRMB15] followed, improving on several aspects of this first model. However, these methods remained either less realistic for strong tissue deformations, required a lot of manual intervention to work or were hard to use in an automatic manner.

The Skinned Multi Person Linear model (SMPL) by Loper et al. [LMR+15] tries to alleviate these issues (see Figure 2.5). It is built from ∼2,000 3D scans of people from the CAESAR dataset [RBD+02]. It is fast to evaluate and differentiate. At the same time, it produces realistic results for a variety of people even in poses with strong tissue deformations. For this purpose, it is using blend shapes for identity, pose, and soft-tissue dynamics. It exposes this functionality in a standard way so that it is usable with all major animation software toolboxes and comes with a full fledged Python interface. This makes the combination with optimization tools like chumpy [Lop15] and OpenDR [LB14] possible, which we use in Chapter 6 to perform gradient descent on an energy function including the SMPL parameters.

The model consists of a mesh with $N = 6{,}890$ vertices and a kinematic tree with $K = 23$ joints. We will treat the body model as a black box, working solely with the posed meshes. A 3D SMPL mesh M is parameterized as $M(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\gamma})$ with pose $\boldsymbol{\theta}$, shape $\boldsymbol{\beta}$, and global translation $\boldsymbol{\gamma}$. $\boldsymbol{\beta}$ is a vector of weights in a Principal Component Analysis (PCA) space of vertex displacements. The vector components measure the distances of vertices of a fitted, artist designed base mesh to the 3D scans of the CAESAR dataset. The identity shape is the weighted sum of PCA components. For both, $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$, separate blend shapes are added to the identity shape with a standard blend skinning function. Even the canonic joint locations of the rig are modeled as a function parameterized on $\boldsymbol{\beta}$ to ensure artifact free rendering.

Part II

Machine Learning

Chapter 3

Induced Entropies for Decision Forest Training

Decision Forests are versatile and robust learners. In particular, they are robust to inputs and outputs in differing orders of magnitudes or differently structured Domain and Image spaces (*e.g.*, mapping planar coordinates to the space of rotation matrices). We use them in Section 7.4.2 to predict joint reliability and in Section 8.3.5 to predict body shape and pose parameters from 2D coordinates. For a brief introduction into notation and theory, see Section 2.1.

In this chapter, we consider the entropy function $H(\boldsymbol{p})$ and its differential counterpart $H[\mathfrak{p}]$ that is used to measure the 'impurity' of a set of samples w.r.t. the target concept. It is used during Decision Forest training to optimize the splitting criterion for a decision node. The steepness of the entropy function determines at which subset size and which level of impurity improvement a split of samples is preferred over others. Hence, an accordingly parameterized family of functions with changing steepness would be well-suited for this task. The generalized Rényi and Tsallis entropies fulfill this criterion. However, due to the use of the computationally expensive log function, they are slow to evaluate. We show that by relaxing two of the four Khinchin-Shannon axioms a new family of entropies, the p-norm-induced entropies, arises. It addresses both of the aforementioned issues: its p parameter determines curvature and for whole values of p, the evaluation is very efficient.

In our analysis, we show that the new family of entropy functions is closely related to existing entropies and heuristics for Decision Forest training. At the same time, the evaluation time of the discrete p-norm-induced entropy is by factors smaller than for the classical Shannon entropy for whole p values (*e.g.*, for ten classes around factor five). In experiments with classification, regression and the recently introduced Hough forests, we analyze the performance of the new entropy. The experiments indicate that the impact of the choice of the entropy function is limited, however can be a simple and useful post-processing step for optimizing Decision Forests for high performance applications.

3.1 Related Work

There are two kinds of related work that will be discussed: on the one hand with respect to the split evaluation functions for Decision Forests and on the other hand with respect to the generalization of entropies.

3.1.1 Split Optimization Criteria for Decision Forests

Yu-Shan Shih did a comprehensive review of splitting criteria for classification trees in [Shi99]. A more recent overview can be found in [MR10].

For regression forests, considerably fewer split optimization criteria are used compared to classification trees. The optimization criteria depend on the loss function that is optimized. For the standard mean-squared error loss, usually the negative sum of variances is maximized (which can be written as $-\mathrm{tr}(\boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ is the covariance matrix of the samples). Alternatively, the Least Absolute Deviation (LAD) from the mean can be used, which is less vulnerable to outliers. For a comprehensive review up to the year 2004, see [Bre04]. Criminisi and Shotton use the differential Shannon entropy to estimate the quality of fit of linear models in decision forests [CS13].

The use of Rényi and Tsallis entropies for decision forest training have been evaluated briefly in [MD08]. The evaluation is done on three datasets with less than 80 samples. As a result, the trees reach a depth of about three. We consider this evaluation as not representative for computer vision and extend it to datasets with up to 74000 samples with trees of depth up to 20. Additionally, we explore the use of the differential versions of these entropies for regression and Hough forests.

3.1.2 The Khinchin-Shannon Axioms

The Khinchin-Shannon axioms (KS) for an entropy functional H over the probabilities $\{p_i\}_{i=1,2,\ldots,w}$ are defined as follows:

(KS1) $H(p_1, p_2, \ldots, p_w)$ is continuous with respect to all of its arguments.

(KS2) H takes its maximum for the uniform distribution $p_i = \frac{1}{w}$, $i = 1, ..., w$.

(KS3) $H(p_1, p_2, \ldots, p_w, 0) = H(p_1, p_2, \ldots, p_w)$.

(KS4) Given two systems described by two probability distributions $\boldsymbol{A}$ and $\boldsymbol{B}$,

$$H(\boldsymbol{A} \cap \boldsymbol{B}) = H(\boldsymbol{A}) + H(\boldsymbol{B}|\boldsymbol{A}),$$

where $H(\boldsymbol{B}|\boldsymbol{A}) = \sum_{i=1}^{w} p_i(\boldsymbol{A}) H(\boldsymbol{B}|\boldsymbol{A} = A_i)$.

They were described independently by Yakovlevich Khinchin [Khi57] and Claude Shannon [SW48] and capture the properties of a function that measures the amount

of information required to describe the state of a system. A system with zero entropy is in a 'pure' state: no information is required to describe 'impurities'. An increasing entropy indicates more impurity.

3.1.3 Generalized Entropies

The *Shannon entropy*, being an important measure in information theory and physics, has been shown to uniquely fulfill the *Khinchin-Shannon axioms*. For a system with W states with probabilities $\boldsymbol{p}$, it is defined[1] as $S(\boldsymbol{p}) = -\sum_{i=1}^{W} p_i \cdot \log_2 p_i$.

The first generalization attempt comes from Alfréd Rényi [Rén61]. His parameterized Rényi entropy is equivalent to the original Shannon entropy for $q = 1$. It is defined as $R_q(\boldsymbol{p}) = \frac{1}{1-q} \log_2 \left(\sum_{i=1}^{W} p_i^q\right)$. Constantino Tsallis developed the Tsallis entropy [Tsa88] for non extensive systems: it is defined as $T_q(\boldsymbol{p}) = \frac{1}{q-1}\left(1 - \sum_{i=1}^{W} p_i^q\right)$ and is equivalent to the Shannon entropy for $q = 1$ as well [Tsa88]. Both relax the additivity axiom KS4. Comparisons to the Shannon, Rényi and Tsallis entropies are included in our experiments. Sharma and Mittal developed the *Sharma-Mittal entropy* [SM75]. Their generalization, and to the best of the authors' knowledge the most recent generalization of *supra-extensive entropy* by Marco Masi [Mas05], are the strongest generalizations so far. Both of these entropies generalize Tsallis and Rényi entropy, and hence do not guarantee additivity. They both have two continuous parameters: this deviates from our aim of developing an easy to optimize entropy for decision forest training.

All of the aforementioned approaches leave the first three axioms untouched and, but the Tsallis entropy, make use of the log function. We will show in the following sections that the system on which decision forests work can well be modeled by the first two axioms and does not require the use of computationally expensive log calculations.

3.2 Induced Entropies for Discrete Systems

KS3 says that an additional state with probability 0 does not change the entropy of the system. This is a possible, but not necessary axiom in the context of decision forests: it is known for the entire forest training what states are consistently possible. Moreover, arguably an entropy violating KS3 might be more appropriate in some contexts than one abiding KS3: a system with three states in a configuration with two equally probable states and one with probability zero might have a lower entropy (be more ordered) than a system with only two equally probable states.

[1] We denote the various entropy functions with different function names as implementations of $H(\boldsymbol{p})$ for the sake of an easier description. Additionally, we denote the entropy parameter for all generalized entropies with q for a more consistent notation.

The distinction becomes philosophical: is a state with probability zero different from 'no state' by definition?

3.2.1 Definition

A function fulfilling KS1 and KS2 must be continuous in all of its arguments, assuming its maximum at the point of equiprobability. A particularly interesting function fulfilling these requirements is the negative sum of absolute distances to the point of equiprobability. It can be parameterized with a power parameter q:

$$\tilde{N}_q(\boldsymbol{p}) = -\sum_{i=1}^{W} \left| p_i - \frac{1}{W} \right|^q . \tag{3.1}$$

KS1 holds for this function, since a norm is by construction uniformly continuous in all of its arguments. KS2 holds as well, because of the positivity and the zero vector property. This holds true for all norm-induced metrics that measure the distance to the point of equiprobability. Hence, p-norms can be used to rephrase Equation 3.1:

$$\tilde{N}_q(\boldsymbol{p}) = -\left\| \boldsymbol{p} - \boldsymbol{e} \right\|_q^q , \tag{3.2}$$

where $\boldsymbol{e}$ is the point of equiprobability with $\forall i : e_i = \frac{1}{W}$. KS1 and KS2 still hold in this case, since taking the power retains the zero vector, positivity and continuity properties of the norm.

However, this function is always ≤ 0 and is equal to zero at the point of equiprobability. This can be adjusted by adding the minimum as an offset, so that the function is 0 at the points of perfect order and otherwise > 0. Defining $\boldsymbol{u}$ as $u_1 = 1$, $u_i = 0 \, \forall i > 1$, this can be defined as

$$\left\| \boldsymbol{u} - \boldsymbol{e} \right\|_q^q . \tag{3.3}$$

Combining Equation 3.2 and Equation 3.3, the following formula describes the induced entropy for discrete systems:

$$N_q(\boldsymbol{p}) = \left\| \boldsymbol{u} - \boldsymbol{e} \right\|_q^q - \left\| \boldsymbol{p} - \boldsymbol{e} \right\|_q^q . \tag{3.4}$$

You can find a comparison of the characteristic plots of the induced entropy and the Shannon, Rényi and Tsallis entropies for a two state system in Figure 3.1. For the induced entropy, it is impossible to recover the Shannon entropy due to the missing log function in its definition. The closest fit (Mean Squared Error) for a two state system is reached for the value $q \approx 2.60068$.

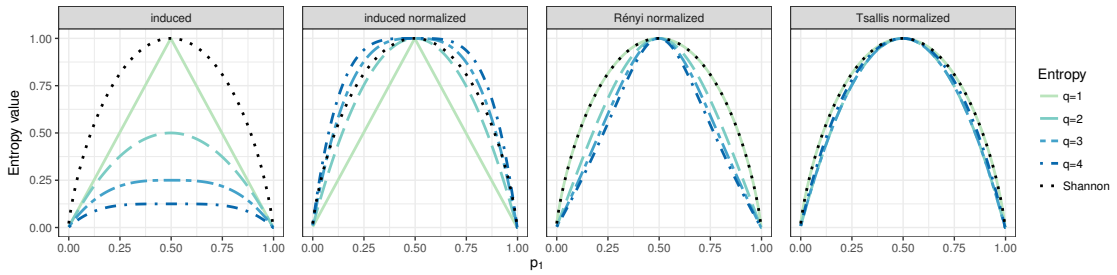


Figure 3.1: *Entropy values for a two state system ($p_2 = 1 - p_1$) for the Shannon entropy and generalized entropies.* The generalized entropies have been maximum normalized except for the induced entropy. The q parameter controls the steepness of the generalized entropies. [LL15a] © 2015 IEEE.

3.2.2 Properties

As described before, KS1 and KS2 hold for N_q. In this section, we will analyze further properties of the introduced entropy.

3.2.2.1 Concavity

For values of $q \geq 1$, N_q is a strictly concave function. Remember that, since it is convex and symmetric, the inducing norm is Schur convex[2] for $q \geq 1$. Since that norm only occurs negative with an exponent ≥ 1 in N_q, the result is a Schur concave function.

For values of $q \in]0; 1[$, the function is still well-defined, however non-concave. For these values, the inducing term loses its property as norm, because the triangle inequality does not hold any more. However, since the term only occurs to the power of q, the resulting function still defines a metric (for more information, see Appendix B).

3.2.2.2 Lesche stability

Lesche stability is claimed to be a necessary condition for an entropy to be a physical quantity [Les82], however this is not undisputed [Yam04]. Proving Lesche stability is a non-trivial task and beyond the scope of this work. We note, however, that the Tsallis entropy has been proven to be Lesche stable by Abe [Abe02] for positive values of q. As we will show in the following Section, the induced entropy is for $q = 2$ equivalent to the Gini measure and T_2, hence at least Lesche stable for this q value.

[2] A function $f : \mathbb{R}^D \to \mathbb{R}$ is Schur convex iff for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^D$: $\boldsymbol{y} \succ_w \boldsymbol{x} \to f(\boldsymbol{y}) > f(\boldsymbol{x})$, where $\boldsymbol{y} \succ_w \boldsymbol{x}$ means that $\boldsymbol{y}$ weakly majorizes $\boldsymbol{x}$, *i.e.*, $\sum_{i=1}^{k} y_i^{\downarrow} \geq \sum_{i=1}^{k} x_i^{\downarrow}$ for $k = 1, \ldots, D$ and $\boldsymbol{x}^{\downarrow}$ denotes a vector with the same components as $\boldsymbol{x}$ but sorted in descending order.

3.2.3 Equivalence of N_2, the Gini Measure and T_2

The Gini measure of a discrete set of probabilities is defined as $\sum_{i=0}^{W} p_i^2$. In the context of Decision Forests, it is used as the entropy-like function $1 - \sum_{i=1}^{W} p_i^2$. In this form it is equivalent to the Tsallis entropy for $q = 2$:

$$T_q(\boldsymbol{p}) = \frac{1}{q-1}\left(1 - \sum_{i=1}^{W} p_i^q\right). \tag{3.5}$$

It can be shown that $N_2(\boldsymbol{p}) = 1 - \sum_{i=0}^{W} p_i^2$ (for a longer version of the proof, see Appendix A):

$$\begin{aligned} N_2(\boldsymbol{p}) &= \|\boldsymbol{u} - \boldsymbol{e}\|_2^2 - \|\boldsymbol{p} - \boldsymbol{e}\|_2^2 \\ &= \left(1 - \frac{1}{W}\right)^2 + (W-1)\left(\frac{1}{W}\right)^2 - \sum_{i=1}^{W}\left|p_i - \frac{1}{W}\right|^2 \\ &= 1 - \frac{1}{W} - \frac{1}{W} + \frac{2}{W}\sum_{i=1}^{W} p_i - \sum_{i=1}^{W} p_i^2 \\ &= 1 - \sum_{i=1}^{W} p_i^2. \end{aligned} \tag{3.6}$$

Hence, for N_2 KS3 holds since it holds for the Tsallis entropy. However, it is easy to find counterexamples for other values of q, *e.g.*, $N_3([0.5, 0.5]) = 0.25$ and $N_3([0.5, 0.5, 0]) \approx 0.3241$.

3.2.4 Relation to the Classification Error

For $q \to 1$ the Tsallis entropy converges to the usual Shannon entropy. The induced entropy converges to a function similar to the classification error. The classification error measures the 'ratio of misclassification' if the most probable system state was predicted for all elements. It is defined as

$$C(\boldsymbol{p}) = 1 - \max_i p_i. \tag{3.7}$$

The induced entropy is proportional to the classification error for $W = 2$ states: $N_1(\boldsymbol{p}) = 2 \cdot C(\boldsymbol{p})$. In general:

$$N_1(\boldsymbol{p}) = 2\frac{W-1}{W} - \sum_{i \,|\, p_i < \frac{1}{W}}\left(p_i - \frac{1}{W}\right) + \sum_{i \,|\, p_i \geq \frac{1}{W}}\left(p_i - \frac{1}{W}\right). \tag{3.8}$$

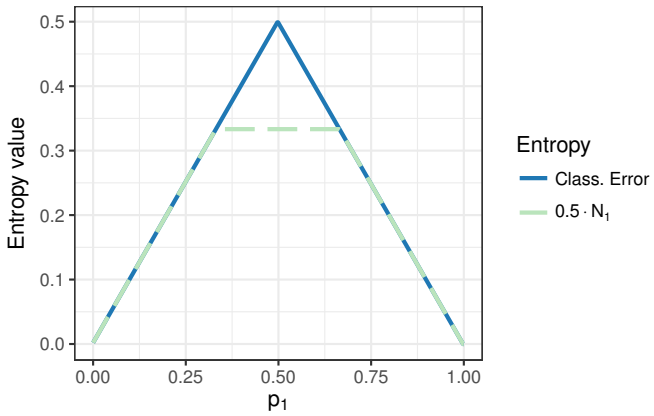


Figure 3.2: *A comparison for classification error and* $0.5 \cdot N_1$ *for a three state system.* p_3 has probability 0 and p_1 and $p_2 = 1 - p_1$ varies. The plateau for the induced entropy is due to constant city-block distance to the point of equiprobability. It is only occurring for N_1. [LL15a] © 2015 IEEE.

For $W = 2$ classes and equiprobability, $N_1(\boldsymbol{p}) = 1$. For $W = 2$ classes and all other cases, each of the two sums runs over one element and it follows:

$$\begin{aligned} N_1(\boldsymbol{p}) &= 1 + \sum_{i \,|\, p_i < \frac{1}{W}} p_i - \sum_{i \,|\, p_i \geq \frac{1}{W}} p_i \\ &= 1 + \left(1 - \max_i p_i\right) - \max_i p_i \\ &= 2 \cdot \left(1 - \max_i p_i\right). \end{aligned} \tag{3.9}$$

For $W > 2$, $N_1(\boldsymbol{p})$ is equal to the classification error as long as only one state has a higher probability than $\frac{1}{W}$. As an example, the characteristic plot for a three state system with the probability of state $p_3 = 0$ is given in Figure 3.2. As long as the system entropy gets closer to the point of equiprobability (until $p_1 = \frac{1}{3}$), the entropy is rising. From that point on, the city block distance to $\boldsymbol{e}$ does not change any more. This results in a constant entropy value. The entropy values decrease again as expected for $p_1 \geq \frac{2}{3}$. This is an interesting, and maybe desired property for some systems. In the context of decision forest training, we recommend to use N_1 only for few classes, but use it in our evaluations for all datasets to give an impression of its performance.

3.2.5 Efficient Implementation

When using entropies for Decision Forest training, they are used inside of a gain calculation function, which is invariant to scaling and shifting. Hence, all of the aforementioned entropy families may be used without these adjustment terms. This value is particularly efficient to evaluate for $q \in \mathbb{N}$ (see Equation 3.1).

3.3 Differential Induced Entropy

The theory introduced so far applies to systems with discrete states. Criminisi and Shotton have proposed to use the differential Shannon entropy to evaluate splits for regression forests [CS13]. Following this idea, we introduce the differential version of the induced entropy to evaluate splits for regression forests.

3.3.1 Definition

Developing the differential version of entropy is mathematically expressed as $W \to \infty$. The first notable property arises when examining the normalization offset:

$$\lim_{W\to\infty} \|\boldsymbol{u} - \boldsymbol{e}\|_q^q = \begin{cases} \infty & \text{for } q \in [0;1[, \\ 2 & \text{for } q = 1, \\ 1 & \text{for } q > 1. \end{cases} \tag{3.10}$$

When examining the rest of the formula, a close correspondence to the Tsallis entropy becomes apparent. With $\lim_{W\to\infty} \frac{1}{W} = 0$, $\tilde{N}_q$ becomes (compare to Equation 3.1):

$$-\int |\mathfrak{p}(x)|^q \, dx = -\int \mathfrak{p}(x)^q dx. \tag{3.11}$$

The full differential induced entropy is thus defined as:

$$N_q[\mathfrak{p}] = \begin{cases} 2 - \int \mathfrak{p}(x)dx = 1 & \text{for } q = 1, \\ 1 - \int \mathfrak{p}(x)^q dx & \text{for } q > 1. \end{cases} \tag{3.12}$$

This is close to the definition of the differential Tsallis entropy:

$$T_q[\mathfrak{p}] = \frac{1}{q-1}\left(1 - \int \mathfrak{p}(x)^q dx\right). \tag{3.13}$$

For $q > 1$, both distributions are equivalent but for the factor $\frac{1}{q-1}$. For $q = 1$, both become uninformative, since $\int \mathfrak{p}(x)dx = 1$. Both differential entropies lead to the selection of the same preferred splits because the constant factor is irrelevant when used inside of the node optimization function.

3.3.2 The Normal Distribution

The most interesting probability distribution, for which the induced entropy will be derived here, is the normal distribution. Assuming $\mathfrak{p}(x) = \mathcal{N}(x; \mu, \sigma)$, the integral

becomes (*c.f.* Appendix C):

$$\int \mathcal{N}(x;\mu,\sigma)^q dx = \frac{1}{\sqrt{q}} \cdot \left(\sqrt{2\pi}\sigma\right)^{-(q-1)}. \tag{3.14}$$

Especially interesting for decision forest induction is the multivariate case. It is required for multivariate regression and for Hough forest induction, since the offset regression is done two-dimensional. The formula for an n-dimensional Gaussian with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ is:

$$\int \mathcal{N}(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma})^q d\boldsymbol{x} = \frac{1}{\sqrt{q^n}} \cdot \left(\left(\sqrt{2\pi}\right)^n \sqrt{|\boldsymbol{\Sigma}|}\right)^{-(q-1)}, \tag{3.15}$$

where $|.|$ denotes the determinant. Summing up, the differential induced entropy for a normal distribution is defined for $q > 1$ as:

$$N_q[\mathcal{N}(\boldsymbol{x},\boldsymbol{\mu},\boldsymbol{\Sigma})] = 1 - \frac{1}{\sqrt{q^n}} \cdot \left(\left(\sqrt{2\pi}\right)^n \sqrt{|\boldsymbol{\Sigma}|}\right)^{-(q-1)}, \tag{3.16}$$

where $|\boldsymbol{\Sigma}|$ is σ^2 in the one-dimensional case.

3.3.3 Efficient Implementation

Similar to the discrete case, a few simplifications can be made for the use of this entropy inside of an arg max function. In this case, it is completely sufficient to use the value

$$\hat{N}_q\left[\mathcal{N}(\boldsymbol{x},\boldsymbol{\mu},\boldsymbol{\Sigma})\right] = -\left|\boldsymbol{\Sigma}\right|^{-(q-1)} \tag{3.17}$$

where $|\boldsymbol{\Sigma}|$ can be replaced by σ^2 in the one-dimensional case. Since usually only the diagonal of the covariance matrix is estimated, this value is again fast and easy to compute, especially for $q \in \mathbb{N} \setminus \{1\}$.

3.4 Experiments

We conducted several experiments for the main application areas of Decision Forests to evaluate how to best apply the generalized entropy families. In each experiment, we did a grid search using cross-validation with the Shannon entropy to determine the decision forest parameters on the training set. The grid contained the following values for all experiments: depth $\{15, 20, 25\}$; feature tests per node $\{7, 10, 15\}$; thresholds tests per feature $\{4, 7, 10\}$ and 100 trees per forest. Each score was then determined by 10 training/testing runs with different random seeds (except for the

Name	Samples	Classes	Features	Test size
chars74k [dCBV09]	74107	62	64	7400 (10%)
g50c [CSZ06]	550	2	50	500 (91%)
letter [BL13]	35000	26	16	8750 (25%)
MNIST [BL13]	70000	10	784	10000 (14%)
USPS [Hul94]	9298	10	256	2007 (22%)

Table 3.1: *Classification dataset characteristics.* [LL15a] © 2015 IEEE.

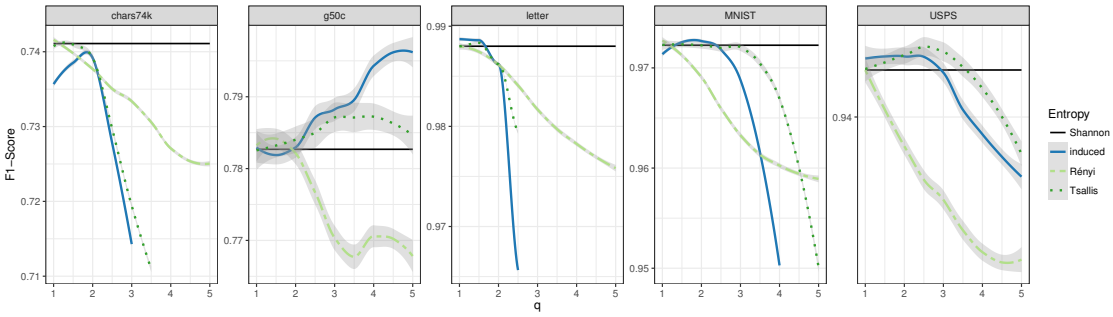


Figure 3.3: *Results of using different entropy functions on the selected classification datasets.* The Shannon entropy has no parameter, hence is always visible as a straight line. Rényi and Tsallis entropy are equivalent to the Shannon entropy for $q = 1$, so all three entropies have the same value at this position. Measurements were taken at each step of 0.5 and are interpolated by applying a Loess smoother, including the 95% confidence interval indicated by a light gray background. [LL15a] © 2015 IEEE.

g50c and *Boston housing* datasets, where due to their small size 250 runs were done). The test set was always selected as by convention for the respective dataset, if available. We designed this setup, since we noticed that the entropy family has none or hardly any effect on other parameters selected by the grid search.

3.4.1 Classification

For the classification setting, we selected five computer vision datasets with varying characteristics. You can find an overview in Table 3.1. Plots of the resulting scores for the standard Shannon entropy and various parameters for Rényi, Tsallis and induced entropy can be found in Figure 3.3. As evaluation measure we used the *F1-score*[3]. It is a reliable measure even for imbalanced datasets, especially when dealing with many classes. Each of the plot facets shows the results for one dataset for each entropy.

Additional to the equivalence of T_1, R_1 and S, the attentive reader may note that the equivalence of T_2 and N_2 is not always given. Investigating on this matter, we

[3] $F_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$, the harmonic mean of precision and recall.

Name	Samples	σ^2	Features	Test size
abalone [BL13]	4177	10.4	9	1044 (25%)
Boston housing [BL13]	506	84.4	14	51 (10%)

Table 3.2: *Regression dataset characteristics.* [LL15a] © 2015 IEEE.

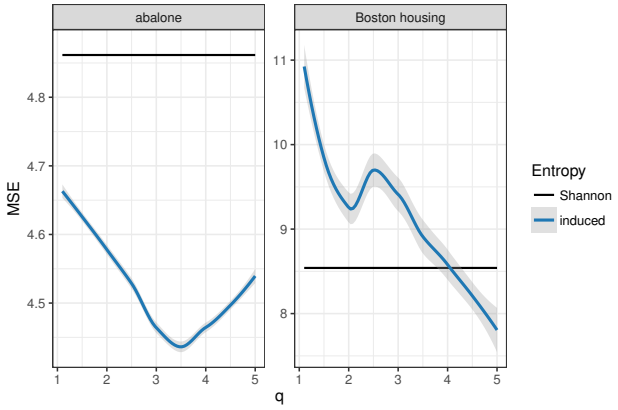


Figure 3.4: *Results on the two regression datasets.* Shannon and Rényi entropy results, and induced and Tsallis entropy results are equivalent in the visualized range. [LL15a] © 2015 IEEE.

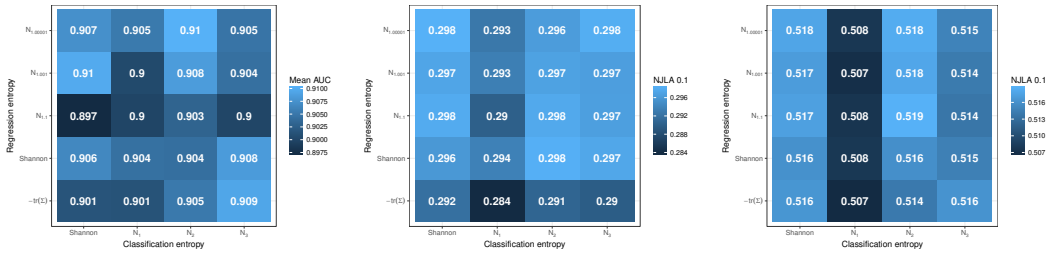
found that the calculation of the entropy values in the two different ways leads to different numerical instabilities (we use `float` values for the calculation of the entropy), that lead to occasionally differently structured trees (approximately 86% of the trees are equal).

The largest effect is observed on the *g50c* dataset with an improvement of about 1.9% when using the induced entropy. Usually, optima are found in the range $q = [1, 3]$ for all entropies. We only observed one exception for the *g50c* dataset, where the Rényi and Tsallis entropies reached slightly higher or comparably high performance results as for the induced entropy for q values close to zero and the results for the Tsallis entropy were still on a high level for $q \geq 6.5$ (but lower than the observed maximum for the induced entropy). On all datasets but *chars74k*, an improvement could be achieved by choosing a different entropy, however, relatively small.

3.4.2 Regression

Regression is, especially in computer vision, not as common as classification. Hence, we selected two non-vision datasets to cover the topic (the characteristics can be found in Table 3.2), and applied the differential entropies in a computer vision setting using Hough forests (see Section 3.4.3).

For regression, we used the Mean Squared Error (MSE) as evaluation measure. You can find the results in Figure 3.4. Rényi and Tsallis entropies are omitted



(a) Weizmann horse dataset. **(b)** LSP dataset. **(c)** FashionPose dataset.

Figure 3.5: *Hough Forest results for several classification and regression entropies.* [LL15a] © 2015 IEEE.

in the plot, since their results are equivalent to the ones of Shannon and induced entropies respectively in the plotted range except for other numerical instabilities. On both datasets, the scores could be improved by using the induced entropy. On the *abalone* dataset, the performance of the Shannon entropy could be outperformed for all values of q. On the *boston housing* dataset, the minimum lies at MSE 7.9 for $q = 3.5$ (this is not visible in the Loess smoothed plot due to outliers at $q = 3$). While the induced entropy is only defined for $q \geq 1$, Rényi and Tsallis entropies are also defined for values in $[0; 1]$. We checked the performance in these areas for both entropies and observed inferior results.

3.4.3 Hough Forests

Hough forests have so far in literature only been used with the regression optimization measure $\mathrm{tr}(\Sigma)$. We provide values with this measure as baseline and evaluate the suggested differential entropies. As evaluation datasets, we use the *Weizmann horse* [BU02] dataset for detection and localization and the *Leeds Sports Pose* (LSP) [JE10] and *FashionPose* [DGLG13] datasets for human pose estimation.

As regression entropy, we use the induced entropy: the results are equivalent to the ones of the Tsallis entropy in the given range ($R_{q\geq 1} = S$). $R_{q<1}, T_{q<1}$ produced comparable or worse results in our former regression experiments. For classification, we use the induced entropy as well. Remember that $N_2 = T_2$, and $R_1 = T_1 = S$. By exploiting these equivalences, we reach the most expressive set of results. To make sure to present conclusive results, we evaluated R_2, R_3 and T_3 on the Weizmann horse dataset and noted worse results than with the corresponding N_q entropies.

3.4.3.1 The Weizmann horse dataset

The Weizmann horse dataset contains images of horses in slightly varying scales. We chose a 'single detection and localization' setting with a Receiver Operator Characteristic (ROC) area under curve evaluation criterion. The forest configuration was set up similar to [GYR+11].

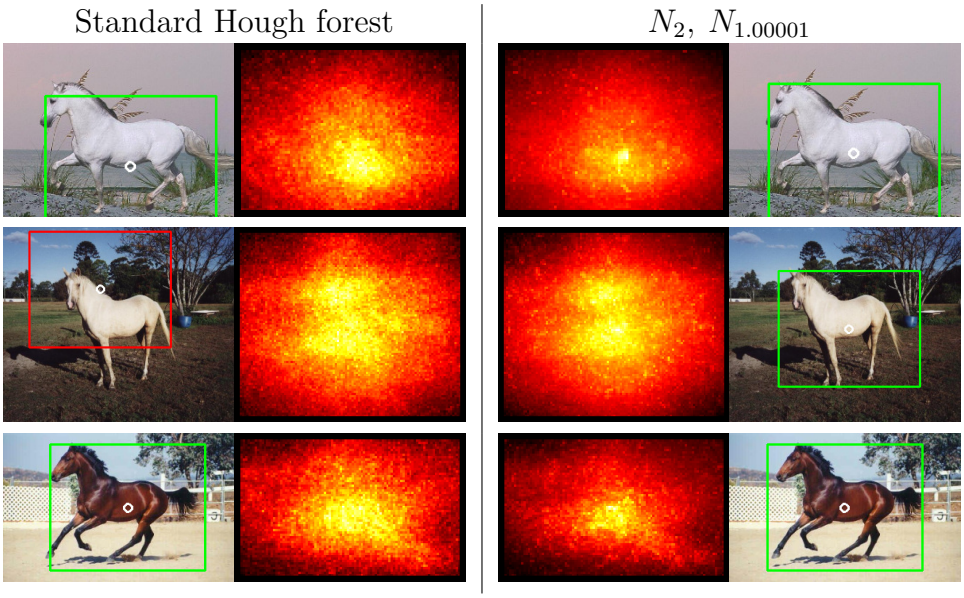


Figure 3.6: *Detections and Hough forest maps on the Weizmann horse dataset [BU02].* q values close to 1 favor larger subsets at splits and are less likely to produce leafs at low levels of the trees. This leads to denser peaks in the predicted probability maps, with an improvement in detection performance. [LL15a] © 2015 IEEE.

Figure 3.5a shows the result matrix. The score for the default Hough forest configuration is located in the lower left corner (0.901). The best scores (0.91) can be reached for two entropy combinations with q values very close to 1 for the regression entropy. We found that for the Hough forest task, q values close to 1 produce the best results. For these values, the norm induced entropy becomes increasingly flat. Flat entropy functions give preference to larger subsets and this seems to be a beneficially factor for Hough forest training. Figure 3.6 shows a comparison of results with the original proposed training and with a run using the induced entropy with the best performing parameters: the resulting maps are denser and reach higher peaks at their maxima. This results in an overall advantage in detection performance.

3.4.3.2 Human Pose Estimation

For the human pose estimation experiments, we largely follow the experimental protocol of [DGL14]. We were able to improve the Hough forest training time of three hours on a 700 CPU cluster reported in [DGL14] to two and a half hours on a 64 CPU cluster using the implementation described in Chapter 4. Since the experiments remained time consuming, we did five training/testing runs for each configuration on both datasets. As evaluation measure we used the *normalized joint localization accuracy*, as introduced in [DGLG13] at the threshold 0.1. This value measures the percentage of correctly localized joints with a slack of up to 0.1 times the upper-body size. Of the three presented methods in [DGLG13], we

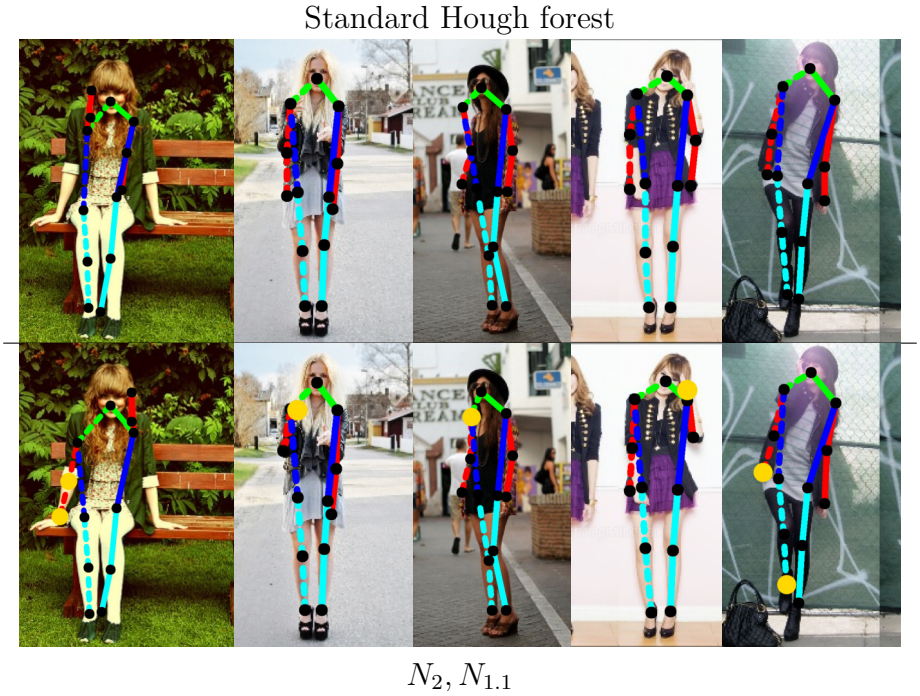


Figure 3.7: *Comparison of Hough forest results on the FashionPose dataset [DGLG13].* Improved joint locations are highlighted with a yellow marker. [LL15a] © 2015 IEEE.

used the *independent joint regression* method to directly show the performance of the Hough forests on the data. We applied a clustered pictorial structure model on the resulting probability maps as described in [DGLG13].

The plots of the results for the two datasets can be found in Figure 3.5b and Figure 3.5c. On the LSP dataset, the performance could be improved from 0.292 to 0.298 for five configurations. On the FashionPose dataset, the score of 0.516 reached by the standard training method could only be slightly improved to 0.519. Across both datasets, we again observed better performance for q values close to 1 for the regression loss and a similar effect of denser peaks in the body part probability maps. Figure 3.7 shows five poses estimated with classical and modified Hough forests for a qualitative comparison. The pictorial structure model profits from the denser probability maps, which mainly results in improved joint localization for the extremities.

3.4.4 Timings

One criterion for the design of the induced entropy function was computational efficiency. We provide a runtime comparison in Table 3.3. The presented norm induced entropy is on par with the Tsallis entropy with a speed gain of more than factor five. At the same time, the induced entropy covers through its equivalences to other entropies an interesting search space for varying values of q. As expected, the classification error is the fastest to evaluate.

Entropy	Shannon	Norm induced	Tsallis / Gini	Rényi	Class. Error
Runtime	1.0	0.19	0.16	0.27	0.12

Table 3.3: *Relative generalized entropy evaluation runtimes compared to the Shannon entropy.* These are timings for discrete entropy and 10 classes, the parameter for the generalized entropies has been set to $q = 3$. All measurements have been performed on an Intel i7-6900K CPU running at 3.2 Ghz.

3.5 Discussion

Relaxing the Khinchin-Shannon axioms to the first two of four, we introduced the new 'induced entropy' in its discrete and differential forms. We analyzed its properties and showed connections to the already established generalized entropies, namely the Shannon, Rényi and Tsallis entropies as well as the heuristic Gini-measure and Classification error.

The new entropy is particularly efficient to evaluate for whole values of q. At the same time, a lot of the entropy search space can be covered with few experiments by exploiting its equivalences to other entropies for certain values of q. Since we noticed that other forest parameters can largely be optimized independently of the entropy type, we suggest to use and optimize the induced entropy as post-processing step after an optimization of forest parameters with the classical Shannon entropy. In our experiments on classification, regression and Hough forests, we noticed minor performance improvements or comparable performance to the Shannon entropy in most settings. In four out of ten experiments, we noticed a strong improvement of 1% or more of the respective evaluation measure just by applying this post-processing step.

Chapter 4

An Object Oriented Decision Forest Implementation

The basic idea of Decision Trees—hierarchical splitting of data with the aim of an increasingly informative grouping—is appealingly simple. On the one hand, this leads to many possible refinements and specializations. On the other hand, many researchers have created their own, hand-tailored implementation for their specific needs. This is especially true for the classic algorithms of the early days of Decision Forests, because each of the algorithms was created for a specific data type and purpose. But if not implemented with thorough testing and modifications for numerical stability and efficiency, even a simple algorithm may turn out erroneous or inefficient.

In their work [CS13], Criminisi and Shotton propose a theoretical model suitable for many tasks and data types. Hence, an implementation of this model could cover a lot of usage scenarios with high re-usability of components. Unfortunately, there is no production ready public implementation available. In this chapter, we propose an object oriented implementation: the 'fertilized forests' library.

With clear interfaces and code locality, the library classes remain easy to understand and easy to alter and extend. While not going as far as to use SSE optimizations, the library is written in templated C++ and optimized thoroughly without destroying code readability. At the same time, OpenMP is used to enable nested parallelization on tree and node optimization level, which allows to make use of many cores even when training fewer trees than available computation cores. The open source library is available under a permissive license with extensive documentation and examples. It provides consistent interfaces to C++, Python and MATLAB. With the idea to be easily extendable, it features an interface generator that automatically keeps the interfaces up-to-date with changes in the core library.

While the library is significantly more flexible than others with similar features, it stays competitive in terms of speed and learning performance (an evaluation is given in Section 4.3). With a focus on computer vision applications, it provides an interface to CAFFE [JSD+14] that can be used for feature extraction.

Name	License	Core lang.	Available bindings	OS ind.	Parallelization	Type aware	Extend.	Setups
fertilized forests	BSD	C++	Python, Matlab	✓	Trees & Nodes	✓	✓	2295
scikit-learn [PVG+11]	BSD	Cython	Python	✓	Trees			32
OpenCV [Bra]	BSD	C++	Python	✓		✓		9
Sherwood [CS13]	MSR-LA	C++	C#		Nodes		✓	5
ALGLIB [Boc]	GPL 2+	C	C#, VB.NET, Python	✓				1
WEKA [HFH+09]	GPL 2	Java	Python	✓				4

Table 4.1: *Overview over Decision Forest libraries.* The column "**Type aware**" refers to whether a library can treat input data in its native data type. The column "**Extend.**" (extendable) refers to whether it is possible without much overhead to extend the implemented algorithms of the library. "**Setups**" summarizes how many different training setups can be realized with the library. [LL15b] © 2015 Association for Computing Machinery, Inc. Reprinted by permission.

4.1 Features

The library tackles many of the software engineering challenges in its domain by using an object oriented model of Decision Forests. This has several advantages:

Local parameterization Instead of having few, over-parameterized factory functions, parameters are local to the objects they are related to. This means that new parameterized objects can easily be created without having to bloat parameter lists with default values.

Recombination It is natural to recombine objects to create new, meaningful forest types. E.g., Hough Forests can be created by reusing regression and classification threshold optimizers and just adding two new objects.

Code encapsulation Semantically close code is automatically grouped together. This makes understanding and extending the code of a class easier, since only the class's interface must be understood to enhance it.

Inheritance Minor modifications to existing algorithms can be created by inheriting from their defining classes. Not the entire functionality must be rewritten, only the most relevant parts.

The library currently contains classes for all classification and regression concepts described in [CSK11], but has been extended with a state-of-the art implementation of Hough Forests, two variants of boosting, several strategies for split optimization and many entropy functions, including the induced entropies described in Chapter 3.

Deterministic, nested parallelism To be able to fully exploit modern manycore architectures, the library supports deterministic, nested parallelism down to node optimization level. While coarse-grained tree-level parallelism is insufficient on modern machines, an additional parallelization step during node optimization offers a second level of fine-grained parallelism. When carefully implemented, race conditions can be avoided while maintaining good parallelization behavior. This guarantees the same, deterministic results independent of the number of used threads.

Templated classes The library has been created with multimedia and computer vision applications in mind. In the age of big data, the sampled signals are getting so large that inflating the data only to adjust its data type becomes permissively wasteful, e.g., increasing the amount of image data by four only to convert from `unsigned byte` to `float`. To avoid this, all library classes are templated with the relevant types of the data they process. To take the hassle away of re-typing the template parameters frequently and to reflect this concept in the non-templated languages MATLAB and Python, a factory object is used that receives the template parameters once and then creates all library objects correctly templated for the user's convenience.

OS independence Being compatible to the gcc, Intel and Microsoft Visual C++ compilers, the library can be used on all major platforms. We use the CMake[1] build system to create robust, platform independent build scripts.

Interfaces The library offers completely consistent interfaces to C++, MATLAB and Python. Convenient updating and complete consistency is ensured by using an easy to use interface generator written in Python that comes with the library.

OS and interface independent persistence By using Boost serialization text archives for persistence across all interfaces, the library objects can be serialized and deserialized across all possible platforms and from within all three supported languages. This allows training trees on a large High Performance Computing (HPC) Linux cluster and then performing analyses, *e.g.*, in Python on Windows.

4.1.1 Comparison with Existing Libraries

There are countless libraries for Decision Forest training available (an overview over the most relevant ones is given in Table 4.1). However, most of them are outdated concerning the supported algorithms and none of them offers a comparably versatile combination of open source code, OS availability, cross platform serialization and number and completeness of interfaces. Curiously, there is no other Decision Forest library with support for MATLAB and the built-in Decision Forest implementation is outdated and slow (more than two orders of magnitude slower than our implementation). It is, however, necessary to note that while a lot of care has been taken to copy as few data as possible in the general library design, a copy of parameters from and to MATLAB is necessary due to its column major storage order concept.

While most of the other software listed as representative selection in Table 4.1 is competitive in the historically relevant areas such as single-threaded runtime and platform availability, few are data type aware, and no other supports nested parallelism and is nearly as versatile and easy to extend. Our aim is to excel in this area by creating a library that is well suited for the dynamic software development cycle of research.

[1] `https://cmake.org`

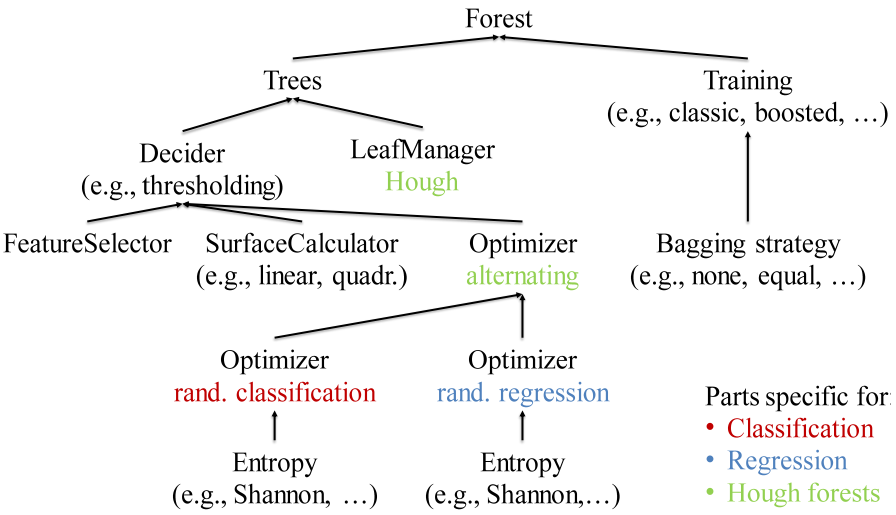


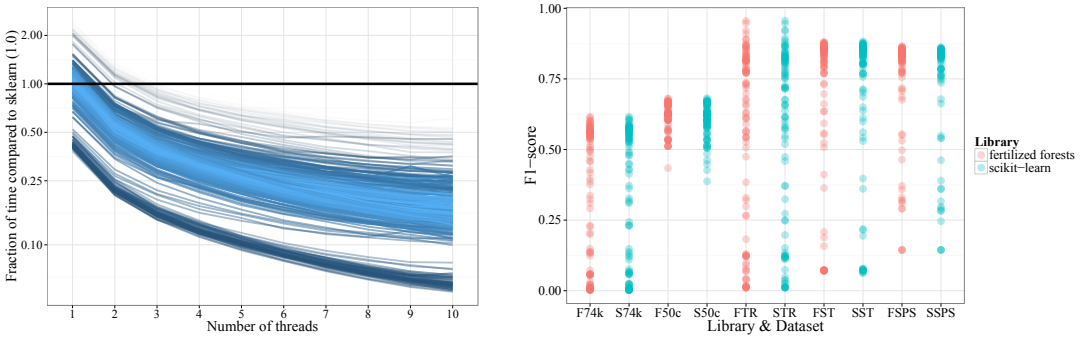
Figure 4.1: *Hough Forest object structure in the fertilized forest library.* Arrows represent 'is-using' relations. Parts of the regular classification and regression forests can be reused to design the Hough Forest specific node optimizer. [LL15b] © 2015 Association for Computing Machinery, Inc. Reprinted by permission.

4.2 Library Design

The library's design is general enough to enable efficient recombination of components and to have benefits by inheritance, but still groups related code together closely. Figure 4.1 shows an overview of the objects necessary to represent a Hough Forest. This specific scenario has been selected, because it illustrates many of the benefits well.

By exchanging the *LeafManager*, which controls the information stored at leafs, as well as the *Optimizer*, which optimizes the thresholds according to the data annotations, it is possible already to change the objective function and the resulting model. This can, *e.g.*, be used to create a classification or regression forest. By adding two new classes, a *HoughLeafManager* and an *AlternatingOptimizer*, and by reusing existing classes, the new concept of a Hough Forest can be defined. It is trivial to integrate the entropies defined in Chapter 3 by creating and using corresponding objects.

Parallelization is implemented in the *Training* (parallelization over tree training) and *Decider* (parallelization over decision optimization) objects. All objects being located lower in the hierarchy than 'Tree' and 'Decider' automatically benefit from these parallelization techniques without explicitly implementing it.



(a) Runtime comparison (logarithmic scale for the time axis).

(b) F1-score results.

Figure 4.2: *Comparison with the scikit-learn forests.* Model scores are prefixed with the first character of the library name (*F* for fertilized forests, *S* for scikit-learn) and the rest denotes: *74k:* chars74k, *50c*: g50c, *TR:* letter, *ST:* MNIST, *SPS:* USPS. [LL15b] © 2015 Association for Computing Machinery, Inc. Reprinted by permission.

4.3 Evaluation

To give the reader an impression of the performance of the library, we did comparison experiments with the widely used 'scikit-learn' machine learning package. We selected that library as competitor, since (a) we appreciate its impact on the machine learning community and its high popularity, (b) used it as inspiration for our project, and (c) the parameterization of this library is similar enough to ours to allow a fair comparison.

Runtime It is not straightforward to set up an experiment resulting in a reliable statement about library speeds: specifics of the data can bias results in favor of each candidate, or a specific parameter setting can be in favor of a specific implementation. Therefore, we performed 100 runs with randomly (but equivalently) sampled parameter settings on each of three large scale, computer vision datasets (chars74k [dCBV09], MNIST [BL13] and USPS [Hul94]). This results in a large set of overall runs under diverse conditions, and we hope that it provides a representative impression of overall performance.

While parallelization over trees trivially has a good parallelization behavior, this is not necessarily true for deterministic parallelization over the node optimization. To visualize the behavior of our library in this specific case, we varied the number of threads for node optimization. The results are visualized in Figure 4.2a. For each run, the corresponding runtime of scikit-learn was used as normalization (hence the straight line at height 1.0). All other lines show the traces of our library. The saturation of the color is higher for traces closer to the mean trace.

The first important results are the performance values for one thread. There are a few runs for certain parameter settings where the runtime of our library goes up to more than twice of the runtime of scikit-learn. The mean, however, is at about 1.0, with the start of a solid trace at little more than a third of the runtime of scikit-learn (the y-axis is logarithmic). For an increasing number of threads, the library shows good parallelization behavior peaking out in the best cases at substantially smaller values than 0.1 even though only 10 threads have been used.

We identified the ability to parallelize beyond tree level to be of critical importance. Dantone et al. report a training time of 3 hours on a 700 CPU cluster for their Hough Forest human pose estimation implementation [DGL14]. We could reproduce their training on only 64 CPUs in 2.5 hours on our 64 CPU infrastructure through full parallelization, even though the forest model only uses 10 trees. This translates to a speed increase of more than an order of magnitude.

Classification performance In Figure 4.2b, we provide a comparison of F1-scores with scikit-learn on five large scale computer vision classification datasets (parameters were again randomly sampled 100 times). The x-axis shows the various datasets, where the left of each pair of columns shows the fertilized forests performance, and the right the scikit-learn performance. The datasets, from left to right, are chars74k [dCBV09], g50c [CSZ06], letter [BL13], MNIST [BL13] and USPS [Hul94].

The y-axis shows the achieved F1-score, since some of the datasets have a high number of classes and are not perfectly balanced. The authors of scikit-learn have implemented various heuristics in addition to the traditional Decision Forest algorithm. We went through the source code and added them to our library additional to our own. This gives the fertilized forests sometimes a slight edge over the scikit-learn implementation (visible, *e.g.*, for the *g50c* dataset).

4.4 Source Code

The library is complemented with a test suite and automatically checked for memory leaks with valgrind[2]. It has been applied in various scenarios for computer vision (Chapter 3) and other learning tasks (see Section 1.5 as well as Chapter 8). The main source of information around the library, including API documentation and examples is the website `http://www.fertilized-forests.org`. The source code is available at `https://github.com/ChrislS/fertilized-forests`.

[2] `http://valgrind.org`

Chapter 5

A Convenient Interface for High-Performance Deep Learning

Whereas Decision Forests have great applications in scenarios with scarce data and where intepretability of decisions is needed, deep learning is one of the biggest success stories in the computer vision domain. For several problems it has led to solutions that even surpass human performance [HZRS15, SHM+16]. For the computer vision community, the *caffe* [JSD+14] framework is a popular and frequently used deep learning framework. It is well-tested and many authors published custom layer implementations and models.

caffe does provide Python bindings for the core features, however the main workflow relies on *Google protocol buffers*[1] (which we will abbreviate to 'protobuf') for configuration, network design and serialization. This has been a pragmatic solution for network layouts so far, but new complex training strategies and networks with hundreds of layers [HZRS16a, HZRS16b] push this interface to its limits.

In this chapter, we describe and propose an interface—*barrista*—for the *caffe* framework, which provides a more powerful, programmatic API with solutions and insights that may be of general interest. Furthermore, we strongly advocate the concept of callbacks (which we refer to as 'monitors' throughout the paper, due to the *monitor* software design pattern). We use monitors as an easy-to-write and easy-to-combine tool to create data loaders, preprocessing, hyperparameter modification, logging, post-processing and visualization.

The features of the described interface can be summarized as follows:

- The *barrista* framework provides full Python access to all *caffe* functionalities except multi-GPU training. Networks can be programmatically created and edited in a convenient way.
- Protobuf object introspection is at the core of the *barrista* library. It enables almost automatic parsing of the basic *caffe* interface and guarantees full compatibility and consistency with the used version of *caffe*. This includes custom-written layers and their parameters, which need to be referenced just by their name. All layer parameters are then inferred automatically.

[1] `https://developers.google.com/protocol-buffers/`

- We maintain full compatibility with *caffe* models. By using the internal protobuf representation, it is possible to load and save all models and prototxt files *caffe* can read and write.
- *barrista* provides a *monitor* interface with many existing monitors for preprocessing, postprocessing, data augmentation, visualization, training and model inspection. Many of them can be executed in parallel to the network forward and backward pass to make the execution efficient.
- This provides a principled separation of responsibilities for steps such as data-preparation, data-feedback (*i.e.*, incorporating training results to influence further training for, *e.g.*, active learning) and visualization.

5.1 Related Work

A full overview of deep learning software is out of the scope of this paper. We restrict the discussion to the most prominent frameworks and interface packages targeting the Python language.

The ***Theano*** package [The16b] with its three interface add-ons *Lasagne* [DSR+15], *Keras*[2] and *blocks* [vMBD+15] is similar to *caffe* and *barrista*. *Theano* is a more general machine learning software compared to *caffe*. Therefore, the wrappers are convenient for the specific use case of deep learning. Whereas *Lasagne* keeps a focus on being a lightweight and close wrapper around *Theano*, *Keras* aims for more generality and offers a backend for *Tensorflow* [AAB+15] as well, which will be discussed in the next paragraph. ***Keras*** includes a convenient fitting method, but does not separate responsibilities as clearly as the framework we propose. It offers the possibility to use callbacks, but these can not influence the training or the provided data as deeply as *barrista* and are not executed in parallel. ***Lasagne*** is a lightweight wrapper that focuses solely on Theano's deep learning components. It requires the user to implement a training loop including preprocessing and monitoring. ***blocks*** is very similar to *barrista* in its aims and offers a callback concept with its *Extensions*.

Tensorflow [AAB+15] is Google's open source deep learning framework and subject to rapid changes. It superseded *caffe* as the most popular deep learning framework for computer vision after we published the described interface for *caffe* and we use it in Chapter 9 for our models. We are using and advocating some of the same patterns proposed in this chapter similarly with the Tensorflow framework.

Mxnet [CLL+15] provides APIs for several languages, including Python. It offers high-level training methods and two separate concepts for callbacks. Plain callbacks do not have access to the network's gradient in contrast to a specific monitor object. However, only one such object can be used for training. Both callback types are not executed in parallel and can not be used for data-preprocessing.

[2] `http://keras.io/`

Chainer [TOHC15] fully leverages the Python software stack and is designed for easy modification. This does not extend over the course of training methods, probably because it is relatively easy to implement these methods. However, re-implementation leads to creating the same preprocessing and optimization code repeatedly, which is what we address with our callback stack.

Neon[3] relies on the Python stack with the focus on runtime. It provides a comprehensive callback system, but not for parallel execution. While *Neon* is fast, its commercial distribution model offers parts as premium content. With open source alternatives available, this software has not found widespread use in the research community.

5.2 Concepts and Design

5.2.1 Separation of Responsibilities

One of the core ideas of the library design is the separation of responsibilities:

The user is responsible for preparing the data. This action is highly dataset specific and can hardly be generalized.

The library offers data augmentation, solver setup, logging, training and prediction. Data augmentation methods (like rotation, flipping) can be shared across learning tasks, therefore this responsibility can be moved to the deep learning library.

We believe that a library should solve repetitive tasks for the user out-of-the-box, but remain configurable. Like in the popular *scikit-learn*[4] package, *barrista* offers many default options with sensible values that can be overridden. For example, input data can be passed as a list to the 'fit' or 'predict' methods of a model, but if the data does not fit into memory, it is straightforward to extend the *CyclingDataMonitor* to process a dataset chunk wise.

[3] `https://github.com/NervanaSystems/neon`
[4] `http://scikit-learn.org`

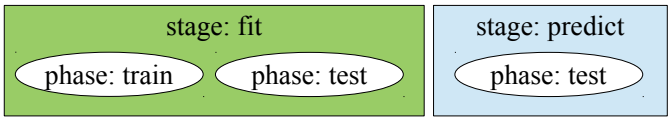


Figure 5.1: *Usage of stages and phases by the 'Net' object.* The *stage* defines the network configuration, the *phase* denotes the current application type. This distinction reflects the underlying *caffe* concepts. If available, the 'predict' stage is used automatically by the 'predict' function. [LKKG16] © 2016 Association for Computing Machinery, Inc. Reprinted by permission.

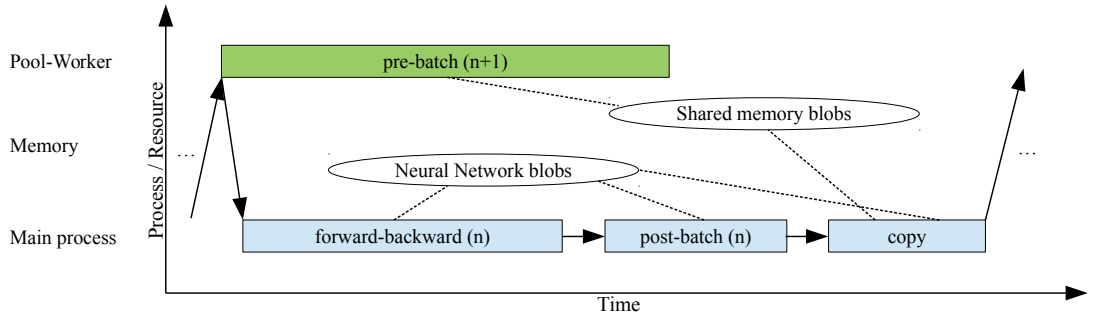


Figure 5.2: *Parallel callback processing pipeline.* Solid arrows indicate control flow, dashed lines resource access. The execution time of the 'forward-backward' and 'pre-batch' operations strongly depends on network layout/pre-processing steps, hence the length of the boxes is not necessarily representative. Memory is only allocated once at the beginning of training and does not change over time. [LKKG16] © 2016 Association for Computing Machinery, Inc. Reprinted by permission.

5.2.2 Representation

One of the strengths of the *caffe* framework is the storage format of models: the network structure is stored separately from network parameters. This has several advantages: most prominently, the structure of the network can be (partially) altered and parameters can still be loaded for the unchanged parts. We aim to reflect this in our interface. The description of a network is stored in a *NetSpecification* object. It can be programmatically constructed, altered, converted to and from a protobuf text representation, the lingua franca for *caffe*. It describes a network *structure* in all possible stages and phases and is independent of model parameters.

To run a network for optimization or inference, its description can be instantiated to create a *Net* object. A *Net* has parameters and can be fitted to data or used to make predictions. If the network description includes a distinction between the 'fit' and 'predict' phases, then the corresponding architectures are automatically generated and used (see Figure 5.1).

In contrast to other frameworks, the *Solver* is represented as a stateful entity, which reflects the underlying *caffe* structure for optimization methods such as Adam [KB14]. Furthermore, this has the advantage that the object can be serialized correctly. All solvers implemented in *caffe* are encapsulated with parameter checks.

5.2.3 Monitoring

Performing research in deep learning requires potential changes at many points and for many objects in the optimization process. Furthermore, it should be easy to adapt, reuse and reorder training components to quickly experiment with new settings. For this purpose, we propose a powerful 'monitoring' system (coined after the 'monitor' software engineering design pattern) with nearly full control over

the training process. Monitors can be used to encapsulate functionality from data loading, preprocessing, optimization, postprocessing and visualization. They can be combined in arbitrary ways. Our experience is that this encapsulation is of tremendous help when exploring the large design space of neural network to quickly experiment with different training strategies.

5.3 Implementation

5.3.1 Protobuf Object Introspection

Building an interface on top of an independently maintained, rapidly developing open source library is challenging. Upstream changes quickly break functionality and an independent project can not rely on support by the main library authors. Even worse: specifying custom *caffe* layers requires changes in the native C++ source code. Hence, many researchers maintain their own fork of *caffe*. If a library working 'on top' of *caffe* would have a hard-coded interface, users would be required to re-export their interfaces one more time. Instead, we propose to use Python's object introspection abilities: all object properties can be queried and their name and type examined. We use this feature to analyze the *caffe*-generated protobuf object. Exploiting the knowledge about *caffe*'s naming convention, we can infer nearly all parts of the interface automatically.

This allows us to work with upstream and custom defined layers with hardly any manual steps. To use a custom defined layer, the only information that must be provided is the relationship between the layer and its parameter protobuf object name, since this is not encoded in the protobuf specification. The rest is inferred by *barrista*. With this architecture, we are able to elegantly avoid the maintenance overhead for wrapping an evolving library. Additionally, we keep the effort for new layer definitions as low as possible. In the core *caffe* library, adding new layers requires code changes in several places.

5.3.2 Monitor Implementation

When implementing the monitors, several points must be taken into account: (1) user designed algorithms must have access to all objects involved in the optimization or training process. (2) it should be easy to recombine existing algorithms. (3) the entire system should be easy to extend for completely novel training strategies that require new types of interactions with algorithms. (4) during the optimization on the GPU, monitors that are independent of the result should run in parallel on the CPU.

To satisfy all these requirements, we use an inheritance-based Signal/Slot design pattern implementation: all user-designed algorithms are derived from a *Monitor*

object. It provides empty implementations for the following 'signal' methods, covering the whole training and inference process:

- initialize_{train, test},
- pre_{fit, test},
- pre_{train, test}_batch,
- post_{train, test}_batch,
- post_{train, test}.

These functions receive their parameters as a dictionary call-by-reference parameter. The *Solver* and *Net* objects are part of this dictionary and can be changed at will. Monitors are registered with the training/inference methods in a list and called according to their order. Like this, the user has full control over execution order and can create arbitrary monitor 'chains'. If a new signal is added to the monitor system, this can be easily done without breaking existing code by adding new, empty methods to the 'Monitor' object.

With these design choices, the system satisfies requirements (1)-(3). If time-consuming operations such as data loading or preprocessing should be performed by monitors without notably slowing down training or inference, they must be performed in parallel during GPU optimizations. The *caffe* data loading facilities can be used for this purpose like any other layer, additionally we provide parallel data layers that are executed in a true parallel context.

Implementing truly parallel systems in Python is challenging due to the global interpreter lock: within one process, only one Python thread may be active at a time. For software that relies on native code execution such as the GPU optimization, this renders inner process parallelism impossible. To circumvent this problem, we rely on the Python *multiprocessing* module to create an additional worker process. Using a separate process can cause latency and performance penalties due to inter process communication. To avoid this overhead, we use shared memory between both processes with locks for synchronization. This functionality is implemented in the *ParallelDataMonitor* object (which inherits from *Monitor*).

Using it is as easy as creating a monitor inheriting from the *ParallelDataMonitor*. Any monitor inheriting from this class will automatically be executed in parallel in the worker thread. The signals remain the same and all parameters can be accessed as for the regular *Monitor*. To achieve this, we use Python's *duck typing*: we create a dummy *Net* object pointing to shared memory with the main process. This has the advantage that any monitor can be used for either serial or parallel execution, *e.g.*, for debugging. After execution of the parallel monitor signals, the main process copies the shared memory to the original *Net* object, an operation that usually takes less than 1ms. A control flow visualization is given in Figure 5.2.

5.3.3 Visualization

With visualization monitors it is possible to create plots in regular intervals and create movies of filter evolution over the course of optimization. The library provides a basic set of monitors for visualization of several quantities: (1) unit activations, (2) gradient histograms, (3) gradient magnitudes and (4) loss values. All of these values help to keep diagnose obstacles for the optimizer, *e.g.*, vanishing gradients in very deep architectures [WRKS16]. Adding even complex visualizations such as the 'guided backprop' visualization [SDBR14] can be easily done by creating additional 'monitors'.

5.3.4 Quality Assurance

To provide high quality code, we follow best practices. With a public CI server that checks primary (tests) and secondary (style) quality of the code, we assure that the library is usable and bugs are detected before a code release. Monitoring the test coverage at the same time (currently more than 94% for the non-user-interface functions of the library; the UI can not be tested on the headless CI server), we make sure that no important parts remain untested. All parts of the library are covered with documentation. Python 3 compatibility is implemented, but is currently not maintainable due to the lacking compatibility of the core *caffe* framework.

5.4 Source Code

The source code is available under the MIT license from Github at `https://github.com/classner/barrista`. It is currently necessary to apply a C++ patch to *caffe* to make *barrista* work. A pull request for the required changes to the master branch of *caffe* is pending (#3629). The change is of general interest, and once accepted we will make *barrista* available as a PyPi project. In the meantime we provide a patched *caffe* version as a submodule of the project. *barrista* is highly flexible and aims to bridge the gap between performance, convenience and continuity. We are confident that it will prove useful in many applications and contribute to an easier usage of *caffe* by reducing development time for both, beginners and experts of deep learning. Many results in the following chapters have been created using *barrista*.

Part III

Combining Human Appearance Models

Chapter 6

Fitting a 3D Body Model to 2D Keypoints

We begin our discussion on combining 2D and 3D human appearance models with the description of a method to automatically fit a 3D body model to 2D joints. The 2D joints carry concentrated information about the basic human pose, hence are a good modality for fitting a 3D model. We fit a generative 3D body model that encodes prior knowledge about the human body and the image formation process. The resulting fit provides us with an estimate about the body shape and 3D pose. Hence, this fitting process forms a cornerstone for the following chapters.

The results described in this chapter originate from a collaborative work with F. Bogo, A. Kanazawa, P. V. Gehler, J. Romero and M. J. Black [BKL+16a]. Since I was contributing to it as second author (first authorship was shared between F. Bogo and A. Kanazawa), I only summarize the ideas in this chapter and refer for a full discussion to our publication [BKL+16a]. My involvement in the project was as follows: I influenced the method design and experiments to integrate 2D keypoint estimation to make the method fully automatic and contributed code for running 2D keypoint pose estimation. I recommended the Leeds Sports Pose (LSP) dataset [JE10] as a basis for experimentation and created a dataset of gender annotations for the dataset to make the method run on it. I ran pose estimation and semantic segmentation as well as detection reliability experiments on the LSP, HumanEva [SBB10] and Human3.6M [IPOS14] datasets, interpreted the results and influenced design choices accordingly.

Previous approaches for 3D human pose estimation usually focus solely on estimating a 3D 'skeleton' (*i.e.*, 3D points and connections) of a human and ignore the body shape. Furthermore, they treat every part of the skeleton independent from all other parts, leading often to implausible limb lengths or body sizes. Instead, we propose to fit the parametric SMPL body model [LMR+15] (for a brief introduction, see Section 2.3) with body shape and skeleton to image evidence. For the basic method described in [BKL+16a] and in this chapter, we focus solely on the usually used 14 pose-defining keypoints as evidence for the fitting process; however, in Chapter 7 we experiment with incorporating foreground information and in Chapter 8 with higher numbers of keypoints including keypoints on the body surface.

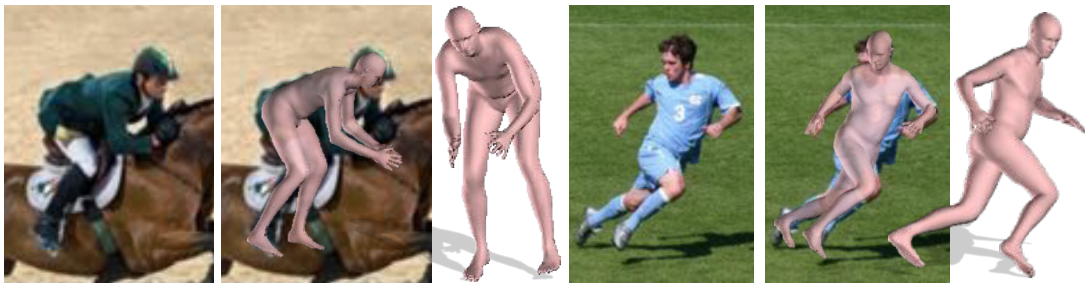


Figure 6.1: *Example results [BKL+16a].* 3D pose and shape estimated using DeepCut CNN [PIT+16] keypoints and the SMPLify energy optimization [BKL+16a] for two images from the Leeds Sports Pose Dataset [JE10]. For each of the two examples: **left:** original image, **middle:** fitted model, **right:** 3D model rendering from a different viewpoint. Springer Proceedings of the European Conference on Computer Vision [BKL+16a] © 2016.

Example results of the fitting process are shown in Figure 6.1. We use the DeepCut CNN [PIT+16] to automatically determine the pose-defining points from an image. Using a carefully crafted energy function, we optimize the pose and shape of the SMPL body model to match the image evidence. Rendering it on the original image shows a good resemblance of 3D body and image evidence; clothing as well as limb and head orientation are not taken into account in the fitting process.

6.1 Method

Our aim is to find a faithful 3D body representation from a single 2D image. We use the DeepCut CNN [PIT+16] to provide a 2D body joint estimate $\hat{\boldsymbol{J}} \in \mathbb{R}^{14\times 2}$ with image coordinates for every joint i as well as confidence values $\boldsymbol{w} = (w_i)$. With the body shape parameters $\boldsymbol{\beta}$, body pose parameters $\boldsymbol{\theta}$, camera parameters $\boldsymbol{K}$ (translation and rotation), we minimize the objective function $E(\boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{K}, \hat{\boldsymbol{J}}, \boldsymbol{w}) =$

$$E_J(\boldsymbol{\beta}, \boldsymbol{\theta}; \boldsymbol{K}, \hat{\boldsymbol{J}}, \boldsymbol{w}) + \lambda_\theta E_\theta(\boldsymbol{\theta}) + \lambda_a E_a(\boldsymbol{\theta}) + \lambda_{sp} E_{sp}(\boldsymbol{\theta}; \boldsymbol{\beta}) + \lambda_\beta E_\beta(\boldsymbol{\beta}), \tag{6.1}$$

consisting of five components with weights $\lambda_\theta, \lambda_a, \lambda_{sp}$ and λ_β. In the following paragraphs, we briefly describe each component.

Joint matching term E_J The joint matching term measures the error in 2D of projected skeleton joints of the body model w.r.t. the estimated 2D joint positions $\hat{\boldsymbol{J}}$. To be robust against outliers during the optimization, we use the differentiable Geman-McClure penalty function ρ [GM87]. With Π_K denoting the perspective projection with camera parameters $\boldsymbol{K}$, we define it as:

$$E_J(\boldsymbol{\beta}, \boldsymbol{\theta}; \boldsymbol{K}, \hat{\boldsymbol{J}}, \boldsymbol{w}) = \sum_{\text{joint } i} \sum_{c \in \{0,1\}} w_i \rho \left(\Pi_K(R_\theta(\boldsymbol{J}(\boldsymbol{\beta})))_{i,c} - \hat{\boldsymbol{J}}_{i,c} \right) \tag{6.2}$$

where $\boldsymbol{J} : \mathbb{R}^B \to \mathbb{R}^{23\times 3}$ is the function that estimates 3D skeleton joint locations from body shape parameters and $R_\theta : \mathbb{R}^{23\times 3} \to \mathbb{R}^{23\times 3}$ is the transformation to pose 3D joints according to pose θ; both functions are differentiable. We use $B = 10$ body shape parameters in our experiments.

Pose prior $\boldsymbol{E_\theta}$ We model the space of plausible poses with a Gaussian Mixture Model (GMM) fitted to the CMU motion capture database [CMU]. This component has a higher energy contribution for less likely poses. For details, we refer to [BKL+16a].

Joint angle prior $\boldsymbol{E_a}$ The computer graphics based SMPL model does not have natural joint limits. This means that arms and legs could be bent 'backwards' if the joint limits are not modeled explicitly. We use the exponential function

$$E_a(\boldsymbol{\theta}) = \sum_{j \in \text{extremities}} \exp(\theta_j) \tag{6.3}$$

for this purpose. Positive angles correspond to backward bends, hence increase the energy contribution of this component drastically. This makes backward bends not impossible but highly unlikely and the function is trivially differentiable.

Interpenetration term $\boldsymbol{E_{sp}}$ Similarly to backward-bent extremities, a pure graphics based model can suffer from interpenetration. To prevent the optimization from converging to poses with interpenetration, we add another term that measures the interpenetrations of spheres approximating the body. The approximation of the body by spheres leads to a differentiable term that can be evaluated efficiently. For details, we again refer to [BKL+16a].

Shape regularizer $\boldsymbol{E_\beta}$ To prevent the optimization from using exotic body shapes to explain pose, we introduce a regularizer for the body shape. Since the shape parameters are weights for PCA components (*c.f.* [LMR+15]), defining a regularizer as

$$E_\beta(\boldsymbol{\beta}) = \boldsymbol{\beta}^T \boldsymbol{\Sigma}_\beta^{-1} \boldsymbol{\beta} \tag{6.4}$$

where $\boldsymbol{\Sigma}_\beta^{-1}$ is the diagonal matrix with the squared singular values from the SMPL training set is a sensible choice.

We optimize for body and camera rotation and translation, body shape and pose but assume a known focal length. Otherwise, the optimization becomes ambiguous and unstable. Because we noticed empirically that an arbitrary but sensible focal length produces plausible results across the LSP dataset with images from diverse capture devices, we use a fixed focal length and decided against introducing additional complexity by estimating the focal length or introducing another prior.

We initialize the camera translation according to the estimated 2D upper body size and position and the pose with a T pose. We then minimize E using Powell's dogleg method [NW99] using OpenDR and Chumpy [Lop15, LB14]. We determined weights for $\lambda_\theta, \lambda_a, \lambda_{sp}$ and λ_β empirically on the HumanEva dataset [SBB10] training part and decrease λ_θ and λ_β over time. Additionally, we start the optimization twice, once rotated by 180 degrees and use the result with the lower energy E_j. The optimization for one image completes in less than 1 minute on a current desktop computer (measurements taken on an Intel i7-6900K CPU running at 3.20GHz).

6.2 Results

We evaluated the presented method in various experiments, including experiments on synthetic data and an ablation study. For a full discussion of results, we refer to [BKL+16a]. We only discuss two results here to give the reader a rough impression of the performance of the algorithm since we build on it in the following chapters.

Table 6.1 provides an overview of results on the Human3.6M dataset [IPOS14]. We obtained frame-by-frame keypoint estimation results by running the DeepCut CNN [PIT+16] on the 15 sequences from subjects S9 and S11 captured from the frontal camera of trial 1. The CNN was *not* finetuned on Human3.6M data, but we observe stable results due to usually easy to parse backgrounds. We then run the optimization of Equation 6.1 on the resulting keypoints and confidences with the gender-specific body models per-frame. We use a regressor to establish correspondences between the recorded motion capture markers and the SMPL body model. With this setup, we achieve a notable improvement over state of the art on single frame 3D body pose and shape estimation on all test sequences.

The Human3.6M dataset consists of recordings in a lab environment with a motion capture rig. This has the advantage that ground truth 3D data can be used for evaluation, but the disadvantage of lacking generality in poses and backgrounds. This is why we also present qualitative results on the LSP dataset [JE10]. It is one of the standard datasets to evaluate 2D human pose estimation and comes with photos of people performing sports activities in highly diverse settings. Ground truth annotations are limited to 2D keypoint positions. You can find qualitative results in Figure 6.2. Apart from the existing gender specific SMPL models (results rendered in skin color) we also experiment with a gender neutral model (results rendered in blue color). In both cases, we observe mostly well-matching results

	Directions	Discussion	Eating	Greeting	Phoning	Photo	Posing	Purchases	Sit
Akhter & Black [AB15]	199b.2	177.6	161.8	197.8	176.2	186.5	195.4	167.3	160.7
Ramakrishna et al. [RKS12]	137.4	149.3	141.6	154.3	157.7	158.9	141.8	158.1	168.6
Zhou et al. [ZZLD15]	99.7	95.8	87.9	116.8	108.3	107.3	93.5	95.3	109.1
SMPLify	**62.0**	**60.2**	**67.8**	**76.5**	**92.1**	**77.0**	**73.0**	**75.3**	**100.3**
	SitDown	Smoking	Waiting	WalkDog	Walk	WalkTogether		Mean	Median
Akhter & Black [AB15]	173.7	177.8	181.9	176.2	198.6	192.7		181.1	158.1
Ramakrishna et al. [RKS12]	175.6	160.4	161.7	150.0	174.8	150.2		157.3	136.8
Zhou et al. [ZZLD15]	137.5	106.0	102.2	106.5	110.4	115.2		106.7	90.0
SMPLify	**137.3**	**83.4**	**77.3**	**79.7**	**86.8**	**81.7**		**82.3**	**69.3**

Table 6.1: *Human 3.6M results [BKL+16a].* 3D joint errors are given in mm. Springer Proceedings of the European Conference on Computer Vision [BKL+16a] © 2016.

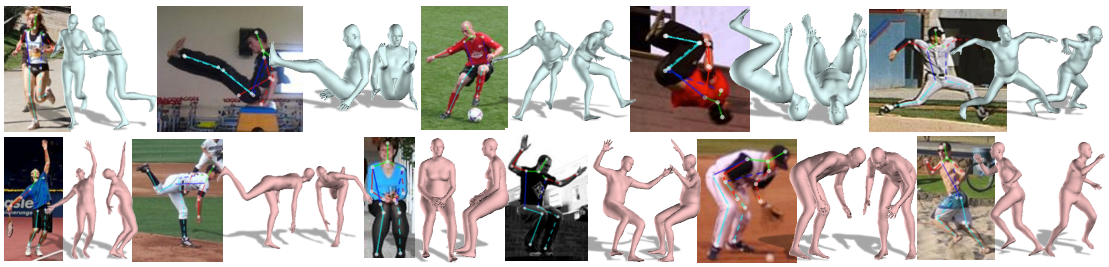


Figure 6.2: *Example results from the* LSP *dataset [BKL+16a].* For each image: **left:** image and DeepCut CNN [PIT+16] keypoints; **center:** 3D fit after optimizing the SMPLify objective function; **right:** rendering of the 3D fit from a different perspective. The **top row** shows example results using the gender neutral model, the **bottom row** shows example results using the gender specific model. Springer Proceedings of the European Conference on Computer Vision [BKL+16a] © 2016.

(15.9 pixel average 2D residuals, *c.f.* Table 7.4). We observe a counter-intuitive effect: the method produces often better results on unusual and intuitively 'harder' poses because the projected 2D keypoint locations are more unique. Intuitively 'easy' poses such as a frontal view of an upright standing person provide little information about the limb length due to the ambiguity between limb length and limb configuration (*e.g.*, for a person viewed from the front with arm joints in a straight line, upper and lower arms could either be shorter or the arm could be bent with the elbow moving further back along the optical axis).

6.3 Discussion

In this chapter, we presented an overview over SMPLify, a fully automatic method for estimating 3D body shape and pose from 2D joints in single images. We use the SMPL 3D body model and fit it to automatically estimated 2D joint locations. The body model implicitly captures correlation between body shape and skeleton parts, hence provides a robust generative model for the fitting process. We present an energy function that has been carefully crafted to avoid local minima during the optimization process, enabling us to use gradient descent based fitting to the estimated 2D joints. With this method, we can fully automatically estimate the 3D body of a person out of a single image with less than a minute runtime on current desktop computers.

Chapter 7

Combining Joints and Segmentation for 3D Fitting

The 3D fitting method presented in Chapter 6 infers body shape from 14 skeleton keypoints well in many cases. The deviations of keypoint locations from a 'standard' skeleton in most views convey enough information to roughly estimate the body shape. However, there are scenarios where this is impossible, *e.g.*, for side views. In these cases, the keypoints collapse to one line, making it merely possible to estimate body height. In these cases, semantic segmentation information helps to find a good shape estimate. In this chapter, we analyze how segmentation information can be used to improve 3D fitting. Furthermore, we show how it can be fused with 2D keypoint predictions to assess their reliability and to improve the segmentation (for an illustrative overview, see Figure 7.1).

For this task, we first build segmentation models that work on a wide variety of challenging poses. While there are several models and datasets for detecting *joints* in challenging images, semantic segmentation is usually considered in more general settings for multiple semantic classes and not focused on humans. Consequently, we annotate the most frequently used human pose datasets with foreground-background and partially with six body part segmentation labels. We use the newly created dataset to train state-of-the art semantic segmentation models for human segmentation. For these models, we report good generalization behavior across datasets and without finetuning.

Since keypoint and segmentation models provide complementary information about a person, we analyze how both can be integrated in several fusion experiments. With fast moving research on human pose estimation as well as semantic segmentation, we do not want to make our fusion strategy dependent on specific predictors. Instead, we treat the predictors as black boxes and suggest fusion methods to work with their predictions. Finally, we extend the energy function developed in Section 6.1 to incorporate segmentation information into the 3D fitting optimization, improving the body shape and pose estimation.

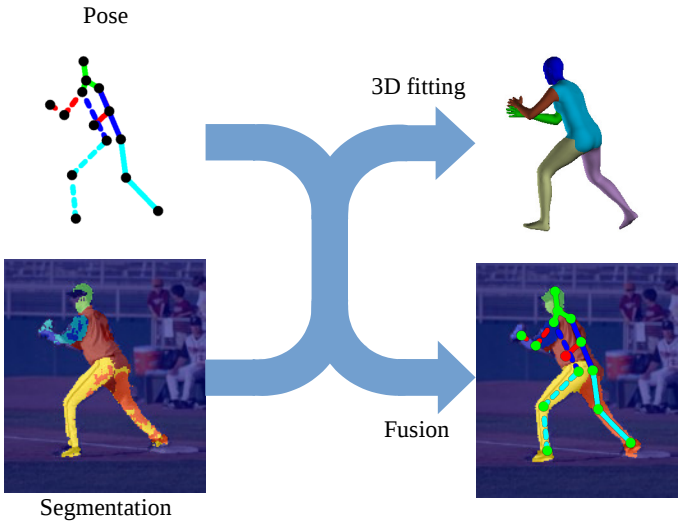


Figure 7.1: *Incorporating segmentation information for fusion with keypoints and 3D fitting.* **Left:** we add segmentation data as additional input besides keypoints to our methods. **Right:** this improves the 3D fitting method developed in Chapter 6 by disambiguating limb configurations and supporting body shape estimation (Section 7.5). Furthermore, we show how keypoints and segmentations can be fused to improve the predicted segmentation maps (Section 7.4.1) and for assessing the reliability of keypoints (Section 7.4.2).

7.1 Related Work

Fitting 3D bodies to silhouettes (*c.f.* [BKL+16a]). Silhouettes intuitively provide strong cues for 3D body pose and shape information. Sigal et al. [SBB08] use the silhouette information to extract local features and use a mixture of experts to directly predict 3D body pose and shape from them. The method assumes that silhouettes are available and we specifically also care for creating them. Guan et al. [GWBB09, Gua12] build a more complex pipeline: they assume manually marked 2D joints, fit a 3D body to them using optimization methods that assume clean ground truth and known segment lengths [Tay00, LC85] and create a foreground silhouette by using GrabCut [RKB04] on the image initialized with the projection of the 3D body. Finally, they fit the SCAPE body model [ASK+05] on a variety of silhouette as well as image cues. This allows to incorporate knowledge from the generative SCAPE model as well as the image evidence into the fitting process. However, the process makes many assumptions and assumes high quality initial data. In contrast, the presented algorithms can handle noisy data and do not require prior knowledge. There are several methods for fitting a 3D body model to silhouettes with significant manual intervention, either with extracting silhouettes manually or specifying point correspondences manually [HAR+10, ZFL+10, CKC10]. In contrast, our methods are fully automatic.

Keypoint and segmentation prediction. In 2004, Mori et al. [MREM04] proposed a method for joined pose estimation and segmentation for upper bodies. The main focus of that work lies on improving human pose estimation by guiding the pose estimation algorithm to the relevant parts of the image. This implements the concept of 'early fusion' (fusing modalities before the reasoning for both has been completed independently) for pose estimation. This requires specifically adjusted models for this purpose and does not allow to treat them as black boxes. Also, they work on a superpixelization of the image and aggregate superpixels, whereas our models work on pixel level. Similarly, Bo et al. perform pedestrian parsing based on superpixelization [BF11]. [SS07] builds a hierarchical parsing of the body where regions are successively combined until a part segmentation is assembled. Lu et al. [LFSL12] start out with coarse segmentation and then predict finer ones, but stop before reaching a level of detail comparable to keypoints.

Several methods formulate a joined energy function to estimate pose and—usually only foreground—segmentation on pixel level [KRBT08, WK11]. [LTZ13] fuses information from a Histogram of oriented Gradients (HoG) [DT05] with a pictorial structure model [FH05] and performs joined energy optimization. These methods are usually computationally intensive since they must solve complex optimization problems.

Foreground background information is of great benefit for tracking methods. Lee et al. [LN09] track people first as segmented foreground blobs before inferring 2D and 3D pose. Ferrari et al. [FMJZ08] use GrabCut [RKB04] to restrict the search space for pose estimation. Bo et al. [BJ13] use dynamic programming to optimize a multi-frame energy that incorporates segmentation information and a pictorial structure model. Similarly, Lim et al. [LHHH13] perform tracking by alternating between optimizing a pictorial structure fit and segmentation.

Pictorial structures [FH00] were very popular for image based inference on human pose and segmentation estimation before the recent advances in deep learning. Since the pictorial structures themselves consist of parts and not only keypoints, they automatically provide a rough segmentation [FH05, ARS09, YR11]. Lu et al. [LSX13] add an explicit segmentation energy that is jointly optimized with the pictorial structure. Another part- and grammar-based pose estimator is described in [SS11]. Rothrock et al. [RPZ13] explicitly describe a grammar and segmentation based model for human pose estimation. A template based model close to pictorial structures taking pose and segmentation into account is described in [CF08]. In contrast to all these models, we do not explicitly enforce structure between the body parts; our learners are entirely data-driven.

There are different notions for 'human parsing'. In [TF10, WTL11], it is understood as part based pose estimation with no explicit segmentation. In contrast, [DCH$^+$16] works with so called 'parselets', image regions originating from segmentation to estimate pose. The segmentation is fashion based, not on anatomical parts. Fusion with an explicit pose estimator is not considered. The idea originates from pose-

lets [BM09], an anatomy based part matching approach not incorporating explicit segmentation information. [YKOB12] focuses on segmenting clothing, taking pose information into account through a Conditional Random Field (CRF) model. We experimented with CRF inference in form of a CNN simulating unrolled inference and message passing in a CRF, but found unrestricted CNN layers to produce superior results.

In contrast to the described methods, we develop fast, discriminative methods to improve and provide pose and segmentation data for 3D fitting. We treat explicit pose and segmentation estimators largely as black boxes. Hence, the proposed methods can be used with many estimators. The only requirement for the estimators is that their performance can be regulated. For our models, we use two strategies for this purpose: early stopping and increasing dropout rates. For most estimators at least one of the two strategies is applicable.

Related datasets. There is a large corpus of work using the *Buffy* dataset [FMJZ08] that provides both pose and segmentation annotations. However, this dataset contains only labels for upper bodies and does not exhibit a high variation in poses or appearances compared to more recent datasets.

The *Human3.6M* dataset [IPOS14] contains video information and 3D pose for 11 different actors and several actions in a lab environment. Segmentation annotation for Human3.6M is available, but was generated using background subtraction which we found to be imprecise. The similar *HumanEva* [SBB10] dataset also provides video information and 3D pose in a controlled environment. Both datasets do not cover appearance and pose variability of datasets "in the wild", hence, we do not find them suitable to train our models. However, we evaluated our segmentation models on both datasets without finetuning to test their generalization ability and present results in Section 7.3.1.

Both, the Pascal VOC Challenge [EGW+10, CML+14] and MSCOCO [LMB+14a] datasets for semantic segmentation include a person class. The coarse polygon annotations and the severe occlusion and truncation of people complicate the matter of a single-person, high accuracy segmentation method. In contrast, we work with pixel accurate segmentation labels. However, we consider to use them to build an extended dataset of people with segmentation, keypoint and 3D body annotation (see Chapter 11).

The *Fashionista* dataset [YKOB12] targets the specific task of parsing human clothing. It collects images from a fashion blog; thus results in limited pose variations. At the same time, *Fashionista* presents a challenging multi-class segmentation problem with different semantics: clothing instead of body parts. The work of [LXS+15] introduces the *Chictopia10k* dataset which builds on the idea of *Fashionista* and adds another 10,000 images to the corpus. However, these datasets do not come with keypoint labels. We test our methods on the datasets to give an impression of the generalization properties of the proposed models and provide results in Section 7.3.

7.2 Segmentation Annotations

Our aim is to train pixel accurate human segmentation models on challenging, real-world images and experiment on the fusion with keypoint models. Since for keypoint estimation, challenging datasets exist we do not start to collect both, keypoint as well as segmentation labels, but augment existing keypoint datasets. For this purpose, we select the Leeds Sports Pose (LSP) [JE10, JE11] dataset and the fully annotated single person instances from the MPII Human Pose Database (MPII-HPDB) [APGS14]. However, annotating at the required level of detail poses big challenges: following the outlines around a person with the mouse takes minutes per image and the datasets consists of ∼25,000 images in total.

To collect segmentation labels on this scale, we built an interactive annotation tool on top of the Opensurfaces package [BUSB13]: Openpose. We make use of the built-in interface with Amazon Mechanical Turk to reach many people as potential annotators. To speed up the annotation process but still work on pixel level, we use the interactive GrabCut algorithm [RKB04]. This breaks down the annotation process in several simple steps: drawing scribbles on the image in either foreground or background. Within few updates, the algorithm finds reliable foreground and background regions that follow the outline accurately. With this interface, an experienced user can segment images in less than 30 seconds. We received many positive comments for the interface. Still, we logged more than 1,000 work hours to obtain human part segmentation for all images (see Table 7.1).

For the LSP dataset, we collected not only foreground background segmentation labels, but also body part labels. The annotated body parts are: head, left and right leg, left and right arm, and torso. This granularity was selected in an attempt to balance the labeling effort and level of detail. To obtain body part segmentation, we use the same binary labeling interface as for foreground background labels but for each part. For these images we collected both, person and body part labels separately. In cases where the part annotation and the foreground annotations don't agree, we used an 'unknown' label. Examples are provided in Figure 7.3.

The segmentation labels capture even fine details of appearance such as the exact outline of pants around the thinner legs. Even though we checked the annotations for every single image and returned particularly badly annotated ones to the pool for labeling, a certain level of noise is unavoidable. This is visible for the lower left arm and the torso labels in the leftmost image. The 'unknown' labels capture uncertain regions well, *e.g.*, in the third image. We experimented with models with and without the 'unknown' labels and note a superior performance if they are used. We publish this dataset together with the 3D annotations presented in Chapter 8 with the 'United People' dataset, hence name the six part annotation dataset *UP-S6h* and the foreground segmentation dataset *UP-S1h* (**U**nited **P**eople - **S**egmentations by **h**umans). To refer to both parts, we use the name *UP-SXh*.

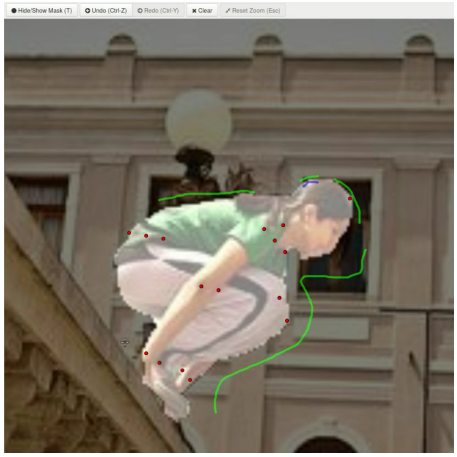


Figure 7.2: *The labeling interface of the Openpose tool.* Green scribbles mark background, blue scribbles foreground. The red dots indicate pose keypoints that are available for every image. We use them to initialize the GrabCut [RKB04] algorithm, which often proposes a good initial solution based on this information.

Dataset	Foreground	6 Body Parts	AMT hours logged
LSP [JE10]	1000 train, 1000 test	1000 train, 1000 test	361h foreground, 131h parts
LSP-extended [JE11]	10000 train	0	
MPII-HP [APGS14]	13030 train, 2622 test	0	729h

Table 7.1: *Datasets and collected annotations.* The average foreground labeling task was solved in 108s on the LSP and 168s on the MPII Human Pose Database (MPII-HPDB) datasets respectively. Annotating the segmentation for one of the six body parts took on average only 40s, all parts 236s.

Figure 7.3: *Examples for six part segmentation ground truth collected with Openpose.* White areas mark inconsistencies between the body part and the foreground segmentation and are ignored. The masks capture the outline of clothing.

7.3 Segmentation Prediction

We use state-of-the art semantic segmentation models as segmentation predictors. We selected the *Deconvnet* [NHH15] model as the highest scoring model with a publicly available implementation on the prestigious VOC 2012 benchmark for the 'human' class at the time of development in 2015. We only slightly deviate from the proposed training procedure: on the UP-SXh datasets, we observed superior performance when training with a batch size of one. This is in contrast to the original method described in [NHH15] where the authors used a higher batch size. During training, the batch normalization layers [IS15] hence receive per batch statistics that can be seen as representative for the full dataset. Consequently, after training, they use the full dataset statistics to robustify the batch norm layers. Instead, we use a batch size of one and per batch statistics for normalization at training and test time. With this method, we obtain superior scores for human segmentation.

7.3.1 Foreground Segmentation

We combine all 25,030 images from our annotated datasets to form a single training corpus. Because we use pretrained pose estimators that have been trained on the full LSP and MPII-HPDB training sets for fusion, we can not use a validation set consisting of training images—the results would not represent true generalization performance. Instead, we use a combined validation set from LSP and MPII-HPDB images, consisting of 50 images from the test set of each dataset. We note that the overlap of the validation set with the test set is small: 5% on the LSP dataset and 1.9% on the MPII-HPDB dataset. Since the validation images are only used for model selection and not training, we believe this setup will hardly influence the test results.

For the Deconvnet-model, we found that a person size of roughly 160 pixels works best for training, therefore we normalize and cut the images accordingly. The images are mirrored and rotated up to 30 degrees in both directions to augment the dataset. We train the network for 300k iterations as determined by the validation set. A summary of scores can be found in Table 7.2, example results in Figure 7.4. On the LSP and MPII-HPDB parts of the UP-S1h dataset, the model consistently achieves accuracy and macro f1 scores higher than 0.9, indicating good performance. Finetuning to the respective dataset parts has only a minor effect on model performance.

Additionally to UP-S1h, we evaluate the model on the three datasets Fashionista [YKOB12], HumanEva [SBB10] and Human3.6M [IPOS14] without finetuning. The images of these datasets are of varying size with people at arbitrary positions in the images. Hence, we use the DeepCut CNN [PIT+16] to extract the people in a preprocessing step. Even though we do not finetune our model to the *Fashionista* dataset, we achieve a competitive accuracy of 0.9738. This speaks for the complexity

	LSP [JE10]	MPII-HPDB [APGS14]	Fashionista [YKOB12]	Human3.6M [IPOS14]
Mean acc.	0.9625	0.9584	0.9738	0.9884
Mean f1	0.9217	0.9092	0.9407	0.8166
Mean acc. (ft.)	0.9684	0.9628		
Mean f1 (ft.)	0.9336	0.9169		

Table 7.2: *Person vs. background segmentation results.* For the result rows with (ft.), the model was finetuned to the respective dataset after training on the full data.

and diversity of UP-S1h. The estimated foreground segmentation accuracy of the current state-of-the art method [LXS+15] is in the interval: $[0.9608; 0.9960]$[1].

We test our model on video data on the *HumanEva* [SBB10] and *Human3.6M* [IPOS14] datasets. The model generalizes well and produces remarkably stable results across frames without smoothing. To calculate accuracy and f1 scores on the Human3.6M dataset, we sample 5 images from all of the commonly used test sequences of subjects 9 and 11, (a total of 1,190 images), and average their scores. The ground truth has been obtained by background subtraction and is rather noisy. This explains the low scores with similarly good looking qualitative results as for the other datasets.

7.3.2 Body Part Segmentation

Because of the more than twice as long annotation time for body part annotations, we collected them only for the 2,000 images of the LSP dataset. This means, there are only 1,000 images available for training which is very little data for a deep learning model. Hence, we initialize the body part segmentation model with the foreground model, drop the last layer and finetune it on the 1,000 part segmentation images for 80k iterations. On average, the model achieves a score of 0.9095 accuracy and 0.6046 macro f1 score (for per part results, see Table 7.3, 'Plain'). Example results can be found in Figure 7.5.

Even with the very limited training data, the model generalizes well in challenging settings. It learns to discriminate between left and right limbs. Also, body parts with only small visible regions are identified correctly (leftmost image). The uncertainty increases in overlapping limb regions or regions far away from the body (rightmost image, left foot). Human annotators label the regions between limbs and upper body as well as shoulders inconsistently. The model predicts usually an average solution, resulting in rounded border areas between limbs and torso.

[1] Only the 56 class accuracy is provided in the paper [LXS+15]: 0.9706. By using the ratio of ∼77% background in the dataset (*c.f.* [YKOB15]), it is possible to give an interval for the foreground performance.

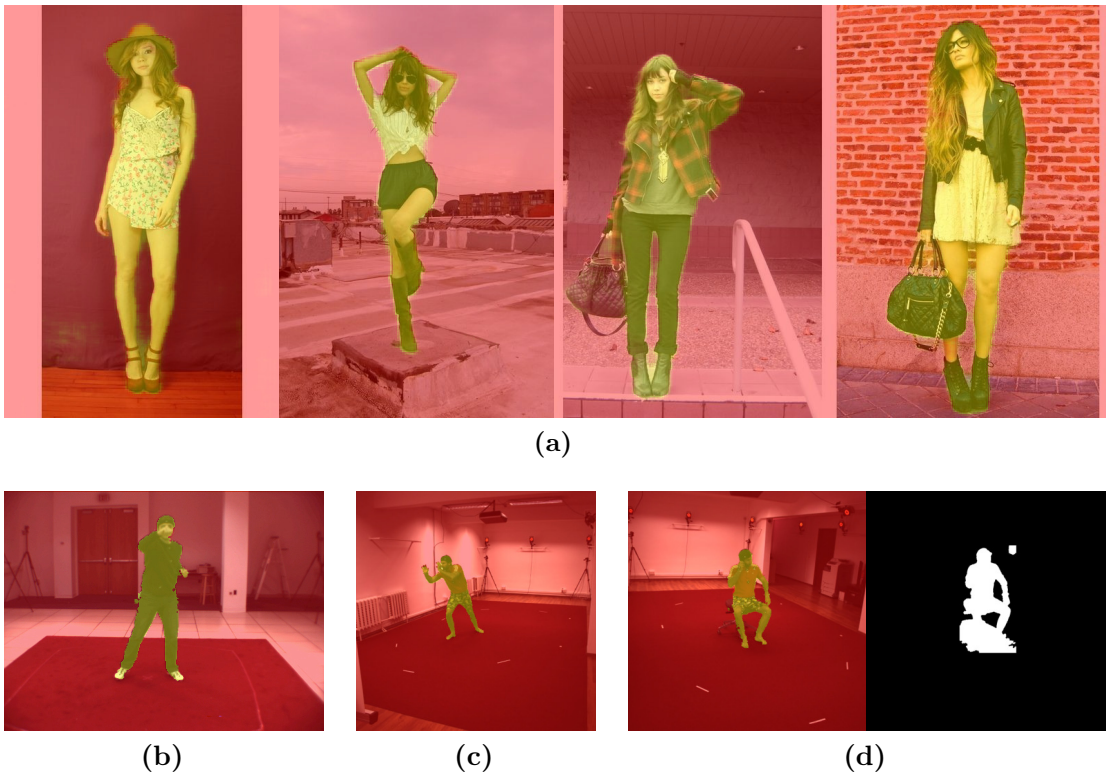


Figure 7.4: *Qualitative foreground segmentation results on various datasets.* **(a)** Results on the Fashionista dataset [YKOB12]. Large handbags are the biggest source of uncertainty without finetuning. **(b), (c)** Results from the HumanEva [SBB10] and Human3.6M [IPOS14] datasets respectively. **(d)** Segmentation ground truth from the Human3.6M dataset is noisy.

Figure 7.5: *Part segmentation results on the* LSP *dataset.* The model learns to discriminate between left and right and detects body parts in small image regions (*e.g.*, leftmost image, right arm).

7.4 Fusion

In this section, we describe two models combining the predictions of the six body part segmentation model with the keypoint predictions of the DeepCut CNN [PIT+16]. Both models are late fusion models, *i.e.*, they work on independently obtained inference results for the input modalities.

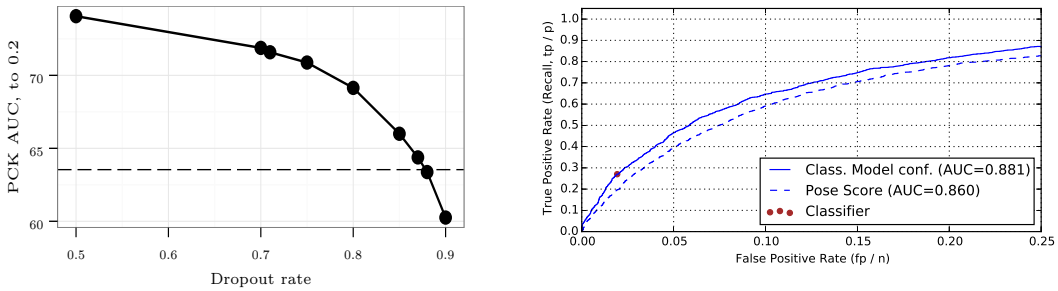
7.4.1 Improving Body Part Segmentation

First, we develop a CNN that receives keypoint and segmentation heatmaps and produces refined segmentation output. We experimented with models for improving both, our foreground segmentation model trained on UP-S1h as well as the six body part segmentation model trained on UP-S6h. The foreground models have already very high performance levels and did hardly benefit of the fusion. In contrast, the body part segmentation task is notably harder and our segmentation models have still a lot of potential to improve. Hence, we use this modality to develop and showcase a fusion model. Whereas we do not use the body part segmentation for 3D fitting in this work—the segmentation term introduced in Section 7.5 only works with foreground information—we note that it would be easy to extend the presented models in that way. Segmentation could be refined up to mesh polygon level and could be very helpful for 3D fitting. We present ideas and preliminary results in this regard in Chapter 11. The fusion methods developed in this section could be used in these scenarios as well.

We treat the inputs to the fusion model as blackboxes, hence work with stacked segmentation and keypoint position heatmaps as inputs. We use a CNN architecture inspired by the multiscale part of the 'fine heat-map' network described in [TGJ+15] to work on the heatmaps, because we found it small in enough in capacity to work with the limited data. At the same time, it works on multiple scales, thus reasoning about small body parts and long range dependencies. After reducing the resolution as described in [TGJ+15], we use fractionally strided convolutions to increase the resolution up to the original image size. We also experimented with a simple stacked conv/relu architecture and refining the network with PReLUs [HZRS15] and batch normalization [IS15], but found the suggested architecture superior (for further information on the hyperparameters, see Appendix D).

The fusion model is trained on the predictions of models that were optimized on the same training data. Because they perform a lot better on their training set than on arbitrary data, *e.g.*, from the test set, we observe that the fusion model does not learn to cope with the lower quality inputs. We identified it to be critical to address this issue to achieve good end results.

We propose two methods to address this issue: increased dropout rates and early stopping. In case of the pose estimation CNN, we only have access to a fully trained model but not the training code. Hence, we increase the dropout rates in several



(a) Performance of the DeepCut CNN on the training set for increasing dropout rates. The dashed line denotes the performance on the test set.

(b) Trust assessment for estimated joint positions (True Positive Rate (TPR) vs. False Positive Rate (FPR)) for plain confidence values and our model.

Figure 7.6: *DeepCut* CNN *dropout performance and trust assessment comparison.*

	Head	Left Arm	Right Arm	Left Leg	Right Leg	Torso	Total
Plain	0.8215	0.4789	0.5300	0.6436	0.6244	**0.7678**	0.6046
Fused	**0.8246**	**0.5714**	**0.6081**	**0.7066**	**0.7100**	0.6922	**0.6342**

Table 7.3: *Comparison of segmentation f1 scores with and without fusion with pose.* The loss in performance for the torso is due to ambiguities of arms in front or back of the torso.

steps until the performance on the training set matches the performance on the test set (see Figure 7.6a). We found this point to be roughly at a rate of 0.88, which is remarkably high and is an indicator for how well the model memorized the training data. Since our segmentation CNN does not use Dropout units but we train it ourselves we use an early stopped state of the model that has similar performance on the training set as the fully trained model on the test set. We use the predictions of these two weaker models to create a representative training dataset for the fusion model.

With this training strategy, we manage to improve the six body part segmentation accuracy from 0.9095 to 0.9155 and the more representative f1 score from 0.6046 to 0.6342. To give an upper bound for the performance of the model, we trained and tested it for fusion with the ground truth pose, which results in 0.9344 accuracy and 0.6667 f1 score. A comparison of per-class scores can be found in Table 7.3; examples are shown in Figure 7.7.

The model uses the pose information to make the segmentation results more coherent, *i.e.*, fills 'holes' in areas of the same body part, and removes uncertain regions with regard to left and right. Segment borders are drawn more clearly. However, the disappointingly low f1 score of 0.6667 for the model trained to fuse segmentation and ground truth pose indicates a design problem: the 2D segmentation and pose

Figure 7.7: *Comparison of six body part segmentation with and without fusion with pose predictions.* For all pairs: **left image**: six body part segmentation; **right image**: estimated pose and fused segmentation. The model fills in holes and missing parts of the limbs where the segmentation is too uncertain. Note the correct segmentation of the right hand behind the body and the correct assignment of torso and leg in the third image.

information is not enough to reason about occlusion, *e.g.*, in a 'torso' labeled region with an arm keypoint, should the fusion model produce an 'arm' or 'torso' labeled result, *i.e.*, is the arm keypoint in foreground or background? This problem has the highest impact for the 'torso' class, which is often occluded by limbs. This is the only class for which the segmentation f1 score decreases through fusion (*c.f.* Table 7.3). This problem can be addressed in many ways, *e.g.*, by adding image evidence or (semi) 3D information as additional input for the fusion model in future work. In the next section, we explore a different possibility for fusing the two modalities that is similarly important for 3D fitting.

7.4.2 Identifying Erroneously Estimated Joints

In this section, we introduce a method to use body part segmentation information to assess the reliability of an estimated joint position. Wrongly estimated positions have a negative influence on 3D fitting performance: they can cause the gradient descent based optimization to get stuck in local minima far away from a sensible solution. The body part segmentation provides additional cues to better identify these joints.

Again, we propose to use a separate predictor working on top of the outputs from pose and segmentation estimators and treating them as black boxes. In this case, we propose to use a Decision Forest classifier working to predict the correctness of estimated joints. For every estimated joint position, we extract several local features that roughly contribute equally to the classification results:

- Body part probabilities for all body parts at this position. This conveys information about left-right uncertainty. Furthermore, frequent error patterns can be detected for certain parts in combination with the other features.

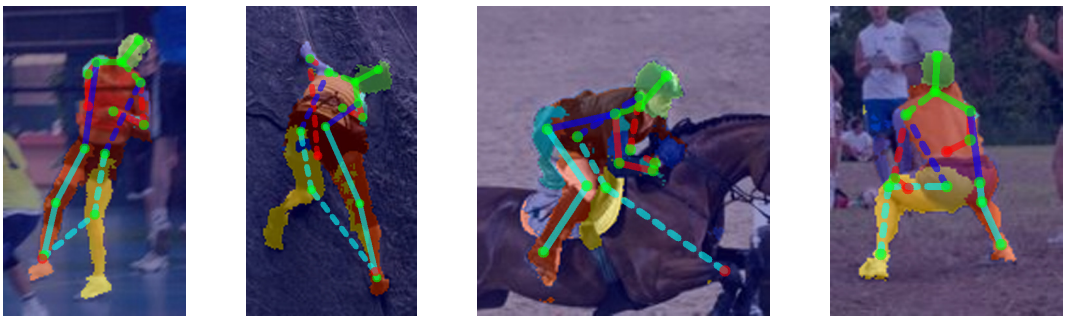


Figure 7.8: *Example predictions of the joint reliability model.* Red marked joints are labeled as 'wrong' by the model. The classical failure case of 'double counting', *e.g.*, both feet are predicted erroneously at the same foot, can be identified often due to the 'background pixels crossed' feature in combination with the body part confidences. The rightmost image shows a failure case where occlusion and low prediction confidences cause (probably) a misclassification.

- Agreement of predicted joint type and body part at this position (*e.g.*, whether the predicted 'left elbow' joint lies within in a region segmented as 'left arm').
- Estimated length of the body part, *i.e.*, the distance(s) to the neighbor(s) in the kinematic tree. This is a strong feature to immediately filter out joints placed far away from the body but with a high confidence score (*e.g.*, Figure 7.8, third image).
- Number of pixels labeled as 'background' on the line connecting the two endpoints of the part the joint belongs to.

With a Decision Forest model trained on these features, we are able to improve the ROC Area Under Curve (AUC) score from 0.860 (decisions based on CNN confidences) to 0.881. A plot of the ROC curves can be found in Figure 7.6b and qualitative examples are given in Figure 7.8. 'Positive' items are wrongly estimated joints, *i.e.*, a true positive rate of 1 means, that all wrongly estimated joints are detected. Especially interesting are areas with a low false positive rate because leaving out a correctly estimated joint position is very disadvantageous for the 3D fit. For the proposed reliability model, the true positive rate is nearly 10 percentage points higher than for the CNN score at a false positive rate of 0.0193. This helps to sort out mistakes of the pose estimator without 'loosing' many correctly detected joints. We use this threshold and estimator for the remaining experiments.

7.5 A Differentiable Segmentation Energy Term

In this section, we present a method to directly integrate segmentation information into the 3D fitting process (this section originates from [LRK+17] © IEEE 2017). For this purpose, we extend the method developed in Chapter 6. In Section 6.1,

Figure 7.9: *Qualitative examples for 3D fit improvements with added segmentation term.* Pose ambiguities due to depth can be resolved better. Rotations of limbs can sometimes be determined due to otherwise unexplained foreground areas (middle image).

we introduced an energy function to fit the pose and shape parameters of a SMPL body model to 2D joints. The objective function is composed of a data term and several penalty terms that represent priors over pose and shape. To fit the 3D model to segmentations of the person, we consequently add one term to the objective to prefer solutions for which the model silhouette and estimated image silhouette, S, match.

Let $M(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\gamma})$ be a 3D mesh generated by a SMPL body model with pose, $\boldsymbol{\theta}$, shape, $\boldsymbol{\beta}$, and global translation, $\boldsymbol{\gamma}$. Let $\Pi(\cdot, \boldsymbol{K})$ be a function that takes a 3D mesh and projects it into the image plane given camera parameters $\boldsymbol{K}$, such that $\hat{\boldsymbol{S}}(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\gamma}) = \Pi(M(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\gamma}))$ represents the silhouette pixels of the model in the image.

We compute the bi-directional distance between $\boldsymbol{S}$ and $\hat{\boldsymbol{S}}(\cdot)$

$$E_S(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\gamma}; \boldsymbol{S}, \boldsymbol{K}) = \sum_{\boldsymbol{x} \in \hat{\boldsymbol{S}}(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\gamma})} \text{dist}(\boldsymbol{x}, \boldsymbol{S})^2 + \sum_{\boldsymbol{x} \in \boldsymbol{S}} \text{dist}(\boldsymbol{x}, \hat{\boldsymbol{S}}(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\gamma})) \tag{7.1}$$

where $\text{dist}(\boldsymbol{x}, \boldsymbol{S})$ denotes the absolute distance from a point $\boldsymbol{x}$ to a silhouette $\boldsymbol{S}$; the distance is zero when the point is inside $\boldsymbol{S}$. The first term computes the distance from points on the projected model to the estimated silhouette, while the second term computes the distance from points in the silhouette to the model. We find that the second term is noisier and use the L1 distance, while the first term uses the L2 distance. We optimize the overall objective with the additional segmentation term using OpenDR [LB14], just as in Chapter 6.

7.6 Evaluation

We use the UP-S6h dataset introduced in Section 7.2 to evaluate the newly introduced methods and their impact on 3D fitting. This dataset gives us not only the possibility to evaluate 2D keypoint error as in Section 6.2, but also to measure how well the body part of the fitted 3D bodies correspond to the human annotated

	2D joint error (train+test)	Seg. acc, f1 (train+test)	2D joint error (test)	Segm. acc., f1 (test)
GT joints	**2.88**	0.9057, 0.7292	**2.91**	0.9033, 0.7231
GT joints, GT seg.	2.91	**0.9116, 0.7424**	2.95	**0.9093, 0.7368**
Est. joints	12.82	0.8904, 0.6841	15.93	0.8853, 0.6680
Est. joints, est. seg.	12.71	0.8955, 0.6932	15.94	0.8908, 0.6772
Reliable est. joints	12.67	0.8900, 0.6817	15.91	0.8857, 0.6676
Rel. est. joints., est. seg.	**12.51**	**0.8970, 0.6955**	**15.71**	**0.8923, 0.6795**

Table 7.4: *Impact of the segmentation term and the joint trust model on 3D fitting performance on the UP-S6h dataset.* The first two rows contain results for ground truth (GT) modalities, the rest of the table uses predictions from the keypoint, segmentation (Section 7.3.1) and reliability models (Section 7.4.2). The 2D joint error is given in average pixel distance. We project the six-part segmented 3D model to the 2D images to calculate segmentation accuracy and f1 score and measure the agreement with the UP-S6h labels.

regions. We then run several 3D fitting trials on ground truth and estimated joints and segmentation and combinations thereof to assess the influence of the proposed methods. A summary of results can be found in Table 7.4, qualitative results are provided in Figure 7.9.

As a baseline, we provide the scores of the SMPLify method running on ground truth joints of the UP-S6h dataset. As expected, the method achieves the overall lowest 2D joint error, since this is the dominating part of its objective function. The f1 score for six body part segmentation is higher than for the CNN predictor, but overall surprisingly low with 0.72 score points when taking into account that the fits could use ground truth information. Clearly, this value is dominated by body parts covering small image regions, such as limbs (close to) perpendicular to the image plane. When adding the segmentation term working with the ground truth segmentation from the LSP part of the UP-S1h dataset, the segmentation f1 score on the test set improves by 1.3 score points, while the 2D joint error only increases by 0.04 pixels, showcasing the potential of this addition. Whereas an improvement of one f1 score point in projected segmentation agreement is not a big numerical difference, it translates to changes in estimated body postures (*c.f.*, Figure 7.9) conveying additional information about limb and body rotations as well as shape. They play a role for applications where exact correspondences between image locations and mesh vertices play a role, *e.g.*, the generation of ground truth for learning tasks as described in Chapter 8.

The second part of the table describes fitting results to estimated joints and segmentation. Line three contains the plain SMPLify result with estimated keypoints. The performance compared to fits to ground truth keypoints remains remarkably high due to the well-performing DeepCut CNN pose estimator. When adding the estimated silhouettes (row 4), the segmentation f1 score increases by nearly one score point while keeping the 2D joint error near constant.

For the last series of experiments we replace the w_i from Equation 6.1 with the output of our reliability model described in Section 7.4.2 to evaluate its effect (row 5). Compared to using plain CNN predictions the average difference is minor: while the method correctly identifies more erroneously estimated joints and excludes them from the fitting process, the 3D fit is unguided for these joints. If the joint is the outer joint of a limb (which is often the case), the optimization will place the limb in the mean pose. Only if an intermediate joint in the kinematic tree is excluded from the optimization, there is hope to find a plausible position conditioned on the position of the rest of the body and its neighbors.

This can be fixed by adding segmentation as an additional cue (row 6): for excluded joints, the segmentation information is used to guide the optimization process. In this scenario, the gain in f1 score points raises to 1.15 (nearly as much as for adding ground truth segmentation information to the ground truth fits to 2D keypoints) and at the same time, the 2D joint error can be reduced (!) by 0.22 pixels.

7.7 Discussion

In this chapter, we described and analyzed multiple techniques to incorporate region information into the 3D fitting process: (1) we created a model that fuses pose information with segmentation information to improve the latter, (2) built a model to assess the reliability of predicted joints and (3) introduced a differentiable error term that integrates segmentation information into the 3D fitting process. To train and evaluate these models, we collected pixel accurate foreground segmentation masks for 25,030 images.

Using the joint reliability model to exclude erroneous joints from the 3D fitting process and incorporating segmentation information to guide it allows us to improve scores of the plain SMPLify method presented in Chapter 6. Whereas the numerical difference between scores is roughly f1 score point, the resulting 3D fits are closer the true 3D pose by slight adjustments in rotation of the body, limbs or the shape. The corrections from incorporating segmentation information into the 3D fitting process are relevant for applications reasoning about vertex correspondences between the 3D mesh and the 2D image locations. This will become important in the next chapter, where we use projected vertices of fitted 3D bodies to create ground truth data for 2D appearance models.

Chapter 8

Increasing the Level of Detail for 2D Models and 3D Fits

Modern CNN algorithms for landmark and segmentation estimation would naturally scale to fine-grained data. At the same time, the 3D fitting techniques introduced in Chapter 6 and Chapter 7 could use more fine-grained 2D cues to produce better 3D results. However, the lack of sufficiently annotated data hinders this development. While it is feasible to annotate a small number of keypoints in images (*e.g.*, 14 in the case of the MPII-HPDB dataset [APGS14]), scaling to larger numbers quickly becomes impractical. The same is true for semantic segmentation annotations: most datasets provide labels for only a few body parts. Annotating with higher granularity is not only cumbersome, it is also prone to annotation inconsistencies.

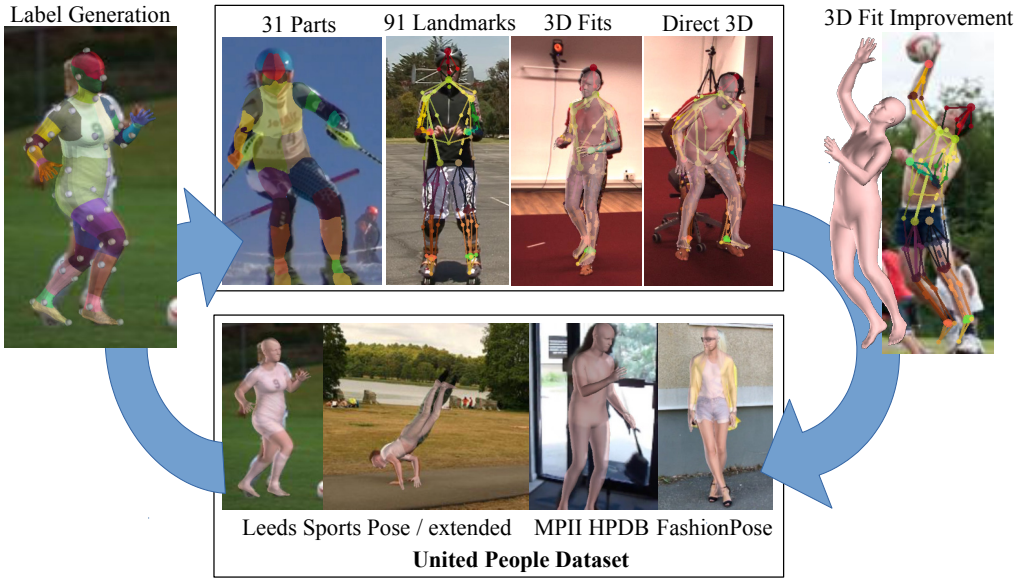


Figure 8.1: *Establishing a virtuous cycle to increase the level of detail for 2D models and 3D fits.* **Lower row:** validated 3D body model fits on various datasets form our initial dataset, *UP-3D*, and provide labels for multiple tasks. **Top row:** we perform experiments on semantic body part segmentation, pose estimation and 3D fitting. Improved 3D fits can enlarge or improve the initial dataset. [LRK+17] © 2017 IEEE.

In this chapter, we develop a self-improving, scalable method that obtains high-quality 3D body model fits for 2D images (see Figure 8.1 for an illustration). To form an initial dataset of 3D body fits, we use an improved version of the SMPLify method described in Chapter 6 that elevates 2D keypoints to a full body model of pose and shape. A more robust initialization and the silhouette objective developed in Chapter 7 help us to apply it on the ground truth keypoints of the standard human pose datasets; human annotators solely sort good and bad fits.

This semi-automatic scheme has several advantages. The required annotation time is greatly reduced (Section 8.2.3). By projecting surfaces (Section 8.3.1) or keypoints (Section 8.3.2) from the fits to the original images, we obtain consistent labels while retaining generalization performance. The rich representation and the flexible fitting process make it easy to integrate datasets with different label sets, *e.g.*, a different set of keypoint locations.

With the labels created from the 3D model we successfully train 2D appearance models with more than factor five more details as they have been usually used so far: a 31 body part segmentation model and a 91 keypoint pose estimation model. Predictions from the 91 keypoint model improve the 3D body fitting method that generated the annotations for training it in the first place. We report state-of-the art results on the HumanEva and Human3.6M datasets (Section 8.3.3). Further, using the 3D body fits, we develop a random forest method for 3D pose estimation that runs orders of magnitudes faster than SMPLify (Section 8.3.5).

Furthermore, the improved predictions from the 91 landmark model increase the ratio of high quality 3D fits on the LSP dataset by 9.3% when compared to the fits using 14 keypoint *ground truth* locations (Section 8.4). This ability for self-improvement together with the possibility to easily integrate new data into the pool make the presented system deployable on large scale.

8.1 Related Work

Acquiring human pose annotations in 3D is a long-standing problem with several attempts from the computer vision as well as the 3D human pose community. The classical 2D representation of humans are 2D keypoints [APGS14, CPM$^+$16, JE10, JE11, ST13, SWT11]. While 2D keypoint prediction has seen considerable progress in the last years and could be considered close to being solved [IPA$^+$16, NYD16, WRKS16], 3D pose estimation from single images remains a challenge [BKL$^+$16a, RKS12, ZFL$^+$10]. Bourdev and Malik [BM09] enhanced the H3D dataset from 20 keypoint annotations for 1,240 people in 2D with relative 3D information as well as 11 annotated body part segments.

In contrast, the HumanEva [SBB10] and Human3.6M [IPOS14] datasets provide very accurate 3D labels: they are both recorded in motion capture environments. Both datasets have high fidelity but contain only a very limited level of diversity in

background and person appearance. We evaluate the 3D human pose estimation performance on both. Recent approaches target 3D pose ground truth from natural scenes, but either rely on vision systems prone to failure [EdAJ+15] or inertial suits that modify the appearance of the body and are prone to motion drift [ZFL+10].

Body representations beyond 3D skeletons have a long history in the computer vision community [Hog83, MN78, NB73, PMTS+15]. More recently, these representations have taken new popularity in approaches that fit detailed surfaces of a body model to images [BKL+16a, GWBB09, HAR+10, LC14, ZFL+10]. These representations are more tightly connected to the physical reality of the human body and the image formation process.

One of the classic problems related to representations of the extent of the body is body part segmentation. Fine-grained part segmentation has been added to the public parts of the VOC dataset [EGW+10] by Chen et al. [CML+14]. Annotations for 24 human body parts and also part segments for all VOC object classes, where applicable, are available. Even though hard to compare, we provide results on the dataset. The Freiburg Sitting People dataset [OVB+16] consists of 200 images with 14 part segmentation and is tailored towards sitting poses. The ideas by Shotton et al. [SGF+13] for 2.5D data inspired our body part representation. Relatively simple methods have proven to achieve good performance in segmentation tasks with "easy" backgrounds like Human80k, a subset of Human3.6M [ICS14].

Following previous work on cardboard people [JBY96] and contour people [FWZB10], an attempt to work towards an intermediate-level person representation is the JHMDB dataset and the related labeling tool [JGZ+13]. It relies on 'puppets' to ease the annotation task, while providing a higher level of detail than solely joint locations.

The attempt to unify representations for human bodies has been made mainly in the context of human kinematics [AAD07, MTD+15]. In this work, a rich representation for 3D motion capture marker sets is used to transfer captures to different targets. The setup of markers to capture not only human motion but also shape has been explored by Loper et al. [LMB14b] for motion capture scenarios. While they optimized the placement of markers for a 12 camera setup, we must ensure that the markers disambiguate pose and shape from a single view. Hence, we use a denser set of markers.

8.2 Building the Initial 3D Dataset

Our motivation to use a common 3D representation is to (1) map many possible representations from a variety of datasets to it, and (2) generate detailed and consistent labels for supervised model training from it. We argue that the use of a full human body model with a prior on shape and pose is necessary: without the visualization possibilities and regularization, it may be impossible to create

sufficiently accurate annotations for small body parts. However, so far, no dataset is available that provides human body model fits on a large variety of images.

To fill this gap, we build on a set of human pose datasets with annotated keypoints. SMPLify (see Chapter 6) produces promising results for automatically translating these into 3D body model fits. This helps us to keep the human involvement to a minimum. With strongly increasing working times and levels of label noise for increasingly complex tasks, this may be a critical decision to successfully create a large dataset of 3D body models.

8.2.1 Improving Body Shape Estimation

When describing the SMPLify method in Chapter 6, we fitted the SMPL [LMR+15] body model to 2D keypoints by minimizing an objective function composed of a data term and several penalty terms that represent priors over pose and shape. However, the connection length between two keypoints is the only indicator that can be used to estimate body shape and limb rotations. Our aim is to match the shape of the body model as accurately as possible to the images, hence we accept the additional computational effort and include the silhouette objective term introduced in Section 7.5 in the objective function. Since we work with ground truth data, the fusion methods developed in Section 7.4 are not required to improve the data quality. However, missing data poses a different, but just as important challenge when fitting to ground truth datasets.

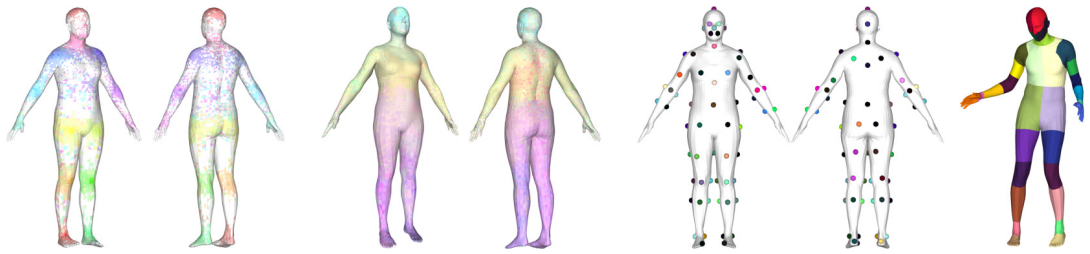
8.2.2 Handling Noisy Ground Truth Keypoints

The SMPLify method is especially vulnerable to missing annotations of the four torso joints: it uses their locations for an initial depth guess, and convergence deteriorates if this guess is of poor quality.

Finding a good depth initialization is particularly hard due to the foreshortening effect of the perspective projection. However, since we know that only a shortening but no lengthening effect can occur, we can find a more reliable person size estimate $\hat{\theta}$ for a skeleton model with k connections:

$$\hat{\theta} = \mathbf{x}_i \cdot \arg\max_y f_i(y), \quad i = \arg\max_{j=1,\ldots,k} \mathbf{x}_j, \tag{8.1}$$

where f_i is the distribution over ratios of person size to the length of connection $\mathbf{x}_i$. Since this is a skewed distribution, we use a corrected mean to find the solution of the arg max function and obtain a person size estimate. This turns out to be a simple, yet robust estimator.



(a) (b) (c) (d)

Figure 8.2: Density of human annotations on high quality body model fits for **(a)** keypoints and **(b)** six part segmentation in front and back views. Areas of the bodies are colored with (1) hue according to part label, and (2) saturation according to frequency of the label. Keypoints on completely 'wrong' body parts are due to self-occlusion. The high concentration of 'head' labels in the nose region originates from the FashionPose dataset, where the 'head' keypoint is placed on the nose. The segmentation data originates solely from the six part segmentation labels on the LSP dataset. (Must be viewed in color.) **(c)** Placement of the 91 landmarks (left: front, right: back). **(d)** Segmentation for generating the 31 part labels. [LRK+17] © 2017 IEEE.

8.2.3 Exploring the Data

With our collected foreground segmentation (*c.f.* Section 7.2) data and the adjustments described in the preceding sections, we fit the SMPL model to a total of 27,652 images of the LSP, LSP-extended, and MPII-HPDB datasets. We use only people marked with the 'single person' flag in the MPII-HPDB dataset to avoid instance segmentation problems. We honor the train/test splits of the datasets and keep images from their test sets in our new, joined test set.

In the next step, human annotators[1] selected the fits where rotation and location of body parts largely match the image evidence. For this task, we provide the original image, as well as four perspectives of renderings of the body. Optionally, annotators can overlay rendering and image. These visualizations help to identify fitting errors quickly and reduce the labeling time to ∼12s per image. The process uncovered many erroneously labeled keypoints where mistakes in the 3D fit were clear to spot, but not obvious in the 2D representation. We excluded head and foot rotation as criteria for the sorting process. There is usually not sufficient information in the original 14 keypoints to estimate them correctly. The resulting ratios of accepted fits can be found in Table 8.1.

Even with the proposed, more robust initialization term, the ratio of accepted fits on the LSP-extended dataset remains the lowest. It has the highest number of missing keypoints of the four datasets, and at the same time the most extreme

[1]For this task, we did not rely on AMT workers, but only on few experts in close collaboration to maintain consistency.

LSP [JE10]	LSP extended [JE11]	MPII-HPDB [APGS14]	FashionPose [DGL14]
45%	12%	25%	23%

Table 8.1: *Percentages of accepted fits per dataset.* The addition of the FashionPose dataset is discussed in Section 8.3.2. [LRK+17] © 2017 IEEE.

viewpoints and poses. On the other hand, the rather high ratio of usable fits on the LSP dataset can be explained with the clean and complete annotations.

The validated fits form our initial dataset with 5,569 training images (of which we use a held-out validation set of 1,112 images in our experiments) and 1,208 test images. We denote this dataset as UPI-3D (**U**nited**P**eople in 3D with an added 'I' for "Initial"). To be able to clearly reference the different label types in the following sections, we add an 'h' to the dataset name when referring to labels from human annotators.

Consistency of Human Labels The set of curated 3D fits allows us to assess the distribution of the human-provided labels by projecting them to the UPI-3D bodies. We did this for both, keypoints and body part segments. Visualizations can be found in Figure 8.2.

While keypoint locations in Figure 8.2a in completely non-matching areas of the body can be explained by self-occlusion, there is a high variance in keypoint locations around joints. It must be taken into account that the keypoints are projected to the body surface, and depending on person shape and body part orientation some variation can be expected. Nevertheless, even for this reduced set of images with very good 3D fits, high variance areas, *e.g.*, around the hip joints, indicate labeling noise.

The visualization in Figure 8.2b shows the density of part types for six part segmentation with the segments head, torso, left and right arms and left and right legs. While the head and lower parts of the extremities resemble distinct colors, the areas converging to brown represent a mixture of part annotations. The brown tone on the torso is a clear indicator for the frequent occlusion by the arms. The area around the hips is showing a smooth transition from torso to leg color, hinting again at varying annotation styles.

8.3 Label Generation and Learning

In a comprehensive series of experiments, we analyze the quality of labels generated from UPI-3D. We focus on labels for well-established tasks, but highlight that the generation possibilities are not limited to them: all types of data that can be extracted from the body model can be used as labels for supervised training. In

the following description of our experiments, we move from surface (segmentation) prediction over 2D- to 3D-pose and shape estimation to a method for predicting 3D body pose and shape directly from 2D landmark positions.

8.3.1 Semantic Body Part Segmentation

We segment the SMPL mesh into 31 regions, following the segmentation into semantic parts introduced in [SGF+13] (for a visualization, see Figure 8.2d). We note that the Kinect tracker works on 2.5D data while our detectors only receive 2D data as input. We deliberately did not make any of our methods for data collection or prediction dependent on 2.5D data to retain generality. This way, we can use it on outdoor images and regular 2D photo datasets. The **S**egmentation dataset UPI-S31 is obtained by projecting the segmented 3D mesh posed on the 6,777 images of UPI-3D.

Following [CYW+16], we optimize a multiscale ResNet101 on a pixel-wise cross entropy loss. We train the network on size-normalized, cutout images, which could in a production system be provided by a person detector. Following best practices for CNN training, we use a validation set to determine the optimal number of training iterations and the person size, which is around 500 pixels. This high resolution allows the CNN to reliably predict small body parts. In this challenging setup, we achieve an Intersection over Union (IoU) score of 0.4432 and an accuracy of 0.9331. Qualitative results on five datasets are shown in Figure 8.3a.

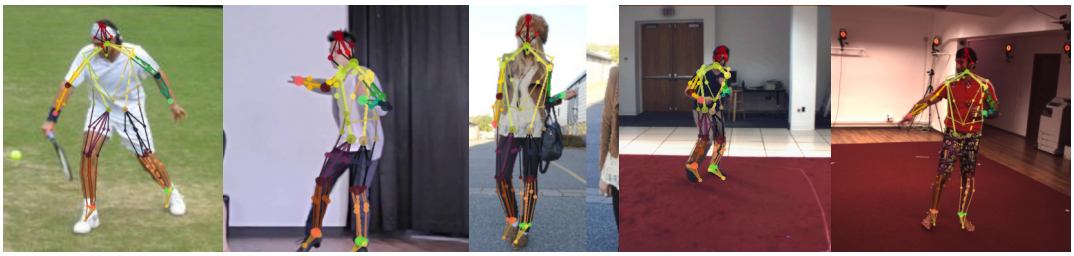
The overall performance is compelling: even the small segments around the joints are recovered reliably. Left and right sides of the subjects are identified correctly, and the four parts of the head provide an estimate of head orientation. The average IoU score is dominated by the small segments, such as the wrists.

The VOC part dataset is a hard match for our predictor: instead of providing instances of people, it consists of entire scenes, and many people are visible at small scale. To provide a comparison, we use the instance annotations from the VOC-Part dataset, cut out samples and reduce the granularity of our segmentation to match the widely used six part representation. Because of the low resolution of many displayed people and extreme perspectives with, *e.g.*, only a face visible, the predictor often only predicts the background class on images not matching our training scheme. Still, we achieve an IoU score of 0.3185 and 0.7208 accuracy over the entire dataset without finetuning.

Additional examples from the LSP, MPII-HPDB, FashionPose, Fashionista, HumanEva and Human3.6M datasets are shown on the project homepage[2]. The model has not been trained on any of the latter three, but the results indicate good generalization behavior. We also present a video to visualize stability across consecutive frames.

[2] `http://up.is.tuebingen.mpg.de/`

(a) 31 Part Semantic Segmentation



(b) 91 Keypoint Pose Estimation

(c) 3D Fitting on 91 Landmarks

(d) Direct 3D Pose and Shape Prediction

Figure 8.3: *Results from various methods trained on labels generated from the UP-3D dataset.*

8.3.2 Human Pose Estimation

With the 3D body fits, we can not only generate consistent keypoints on the human skeleton but also on the body surface. For the experiments in the rest of this paper, we designed a 91-landmark[3] set to analyze a dense keypoint set.

We distributed the landmarks according to two criteria: disambiguation of body part configuration and estimation of body shape. The former requires placement of markers around joints to get a good estimation of their configuration. To satisfy the latter, we place landmarks in regular intervals around the body to get an estimate of spatial extent independent of the viewpoint. We visualize our selection in Figure 8.2c and example predictions in Figure 8.3b.

In the visualization of predictions, we show a subset of the 91 landmarks and only partially connect the displayed ones for better interpretability. The core 14 keypoints describing the human skeleton are part of our selection to describe the fundamental pose and maintain comparability with existing methods.

We use a state-of-the-art DeeperCut CNN [IPA+16] for our pose-related experiments, but believe that using other models such as Convolutional Pose Machines [WRKS16] or Stacked Hourglass Networks [NYD16] would lead to similar findings.

To assess the influence of the quality of our data and the difference of the loss function for 91 and 14 keypoints, we train multiple CNNs: (1) using all human labels but on our (smaller) dataset for 14 keypoints (UPI-P14h) and (2) on the dense 91 landmarks from projections of the SMPL mesh (UPI-P91). Again, models are trained on size-normalized crops with cross-validated parameters. We include the performance of the original DeeperCut CNN, which has been trained on the full LSP, LSP-extended and MPII-HPDB datasets (in total more than 52,000 people) in the comparison with the models being trained on our data (in total 5,569 people). The results are summarized in Table 8.2.

Even though the size of the dataset is reduced by nearly an order of magnitude, we maintain high performance compared to the original DeeperCut CNN. Comparing the two models trained on the same amount of data, we find that the model trained on the 91 landmarks from the SMPL data has a notable advantage of nearly six score points on the SMPL labeled data (rows 2 vs. 3, column 2). Even when evaluating on the human labeled data, it maintains an advantage of two score points (rows 2 vs. 3, column 1). This shows that the synthetic keypoints generalize to the human labels, which we take as an encouraging result.

We provide the third column for giving an impression of the performance of the additional 77 landmarks. When including the additional landmarks in the evaluation, the score rises compared to evaluating on the 14 core keypoints, indicating their overall performance is above average. A direct comparison to the 14 keypoint values is not valid, because the score is averaged over results of different 'difficulty'.

[3]We use the term 'landmark' to refer to keypoints on the mesh surface to emphasize the difference to the so-far used term 'joints' for keypoints located inside of the body.

PCK@0.2	UPI-P14h	UPI-P14	UPI-P91
DeeperCut CNN [IPA+16]	**93.45**	92.16	NA
Ours (trained on UPI-P14h)	89.11	87.36	NA
Ours (trained on UPI-P91)	**91.15**	**93.24**	**93.54**

Table 8.2: *Pose estimation results.* Even though the DeeperCut CNN has been trained on almost by factor ten more examples, our model remains competitive. The third row shows the results of our 91 landmark model evaluated on the 14 core keypoints on human, 14 and 91 SMPL generated landmark labels. It outperforms the model trained on the data labeled by humans (row 2 vs. 3, column 1) by more than two score points. Fair comparisons can only be made within offset boxes. [LRK+17] © 2017 IEEE.

Integrating a Dataset with a Different Label Set The current state-of-the art pose estimators benefit from training on all human pose estimation datasets with a similar label set. The FashionPose dataset [DGL14] would complement these well, but is annotated with a different set of keypoints: the neck joint is missing and the top head keypoint is replaced by the nose. Due to this difference, the dataset is usually not included in pose estimator training.

Using our framework, we can overcome this difficulty: we adjust the fitting objective by adding the nose to and removing the top-head keypoint from the objective function. We fit the SMPL model to the FashionPose dataset and curate the fits. The additional data enlarges our training set by 1,557 images and test set by 181 images. This forms the full UP-3D dataset, which we use for all remaining experiments.

We train an estimator on the landmarks projected from the full UP-3D dataset. This estimator outperforms the plain DeeperCut CNN with a small margin from 0.897 PCK@0.2 (DeeperCut) to 0.9028 PCK@0.2 (ours) on the full, human labeled FashionPose test set.

8.3.3 3D Human Pose Estimation

In this section, we analyze the impact of using the 91 predicted keypoints instead of 14 for the SMPLify 3D fitting method. For the fitting process, we rely solely on the 91 predicted 2D landmarks and no additional segmentation or gender information (in contrast to the SMPLify method as described in Chapter 6 where gender information is used for fitting on the 3D datasets). Segmentation information is not required anymore to estimate body extent due to the landmarks on the body surface.

LSP dataset On the LSP dataset, there is no ground truth for 3D body model fitting available. To be independent of biases towards a specific keypoint set, we

	FB Seg. acc., f1	P Seg acc., f1
SMPLify on GT lms.	0.9176, 0.8811	0.8798, 0.6584
SMPLify on GT lms. & GT seg.	**0.9217, 0.8823**	**0.8882, 0.6703**
SMPLify on DeepCut CNN lms. [BKL+16a]	**0.9189, 0.8807**	**0.8771, 0.6398**
SMPLify on our CNN lms., tr. UPI-P14h	0.8944, 0.8401	0.8537, 0.5762
SMPLify on our CNN lms., tr. UP-P14	0.8952, 0.8475	0.8588, 0.5798
SMPLify on our CNN lms., tr. UP-P91	**0.9099, 0.8619**	**0.8732, 0.6164**
DP from 14 landmarks	0.8649, 0.7915	0.8223, 0.4957
DP from 91 landmarks	0.8666, 0.7993	0.8232, 0.5102
DP from 14 lms., rotation opt.	0.8742, 0.8102	0.8329, 0.5222
DP from 91 lms., rotation opt.	**0.8772, 0.8156**	**0.8351, 0.5304**

Table 8.3: *Scores of projected body parts of the fitted SMPL model on the full* LSP *test set six part human labels (landmarks is abbreviated to lms.).* Fair comparisons can only be made within offset boxes. 'DP' refers to 'Direct Prediction' (see Section 8.3.5). The landmarks for these experiments are always predictions from our CNN trained on UP-P91. [LRK+17] © 2017 IEEE.

rely on the acquired six body part segmentation to obtain meaningful performance scores (see Table 8.3).

The six-part manual segmentation annotations consist of head, torso, left and right leg, and left and right arm (see Figure 7.5). While this representation is coarse, it provides a good estimate of the overall quality of a fit. It takes into account the body shape and not only keypoints, hence it is a fair judge for pose estimators aiming for slightly different keypoint locations.

Unsurprisingly, the segmentation scores of the SMPLify method improve when the segmentation term (*c.f.* Section 7.5 and Section 8.2.1) is added. Due to the longer-trained pose estimator, SMPLify as presented in Chapter 6 still has an overall advantage on the LSP dataset (compare rows three and four).

Training on our generated data for 14 joints and then using SMPLify improves the scores (compare lines four and five) thanks to cleaner data and better correspondence of keypoints and SMPL skeleton. Using our 91 landmark model gives a large performance boost of 3.6 f1 score points. We do not reach the performance of the fits performed on the DeepCut CNN [PIT+16] predictions, largely because of few extreme poses that our pose estimator misses with a large influence on the final average score.

	HumanEva	Human3.6M
Zhou et al. [ZZLD15]	110.0	106.7
DP from 91 landmarks	93.5	93.9
SMPLify on DeepCut CNN lms. [BKL+16a], Chapter 6	79.9	82.3
SMPLify on our CNN lms., tr. UPI-P14h	81.1	96.4
SMPLify on our CNN lms., tr. UP-P14	79.4	90.9
SMPLify on our CNN lms., tr. UP-P91	**74.5**	**80.7**

Table 8.4: *Average error over all joints in 3D distance (mm).* [LRK+17] © 2017 IEEE.

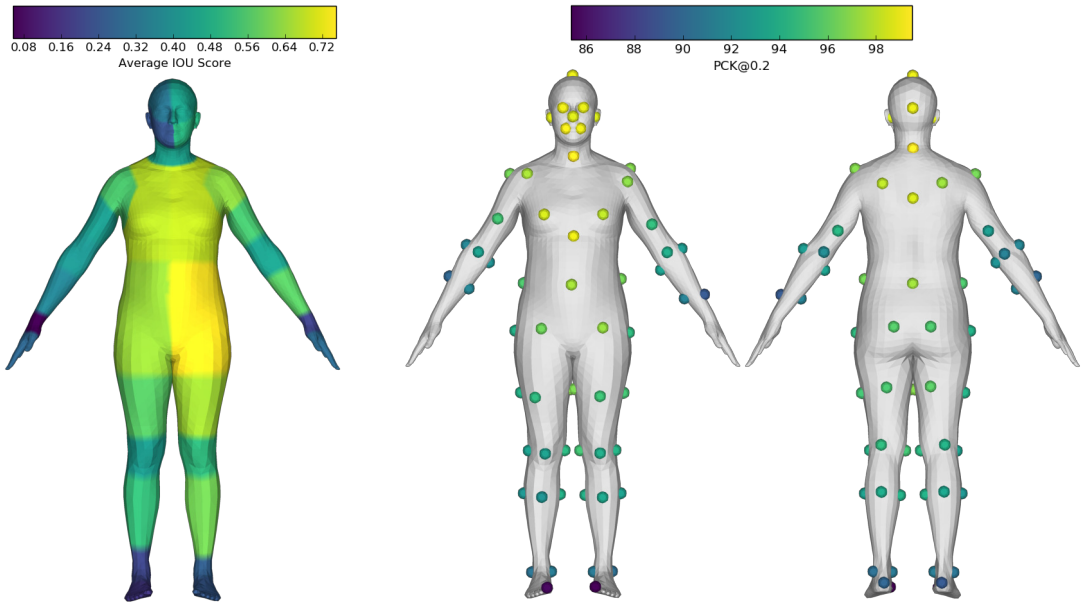
HumanEva and Human3.6M Datasets We evaluate 3D fitting on the HumanEva and Human3.6M datasets where 3D ground truth keypoints are available from a motion capture system. We follow the evaluation protocol of SMPLify to maintain comparability, except for subsampling to every 5th frame. This still leaves us with a framerate of 10Hz which does not influence the scores. We do this solely due to practical considerations, since the SMPLify fitting to 91 landmarks can take up to twice as long as fitting to 14 keypoints. We provide a summary of results in Table 8.4.

We do not use actor or gender specific body models, but one hybrid human model, and rely on the additional landmarks for shape inference. This makes the approach fully automatic and deployable to any sequence without prior knowledge. Even with these simplifications and a magnitude fewer training examples for our pose estimator, we achieve an improvement of 5.4mm on average on the HumanEva dataset and an improvement of 1.6mm on average on the Human3.6M dataset (4th versus 5th row).

The use of a pose estimator trained on the full 91 keypoint dataset UP-P91 improves SMPLify even more. Compared to the baseline model trained on UPI-P14h, performance improves by 6.6mm on the simpler HumanEva dataset, and by 15.7mm on Human3.6M. Even when training a 14 keypoint pose estimator, the higher consistency of our generated labels helps to solve this task, which becomes apparent comparing lines four and five.

8.3.4 Part-by-part Evaluation

In Figure 8.4, we provide visualizations of the scores for fine-grained segmentation and keypoint localization. All the values are from models trained and tested on the full UP dataset. Unsurprisingly, the segmentation scores for wrists, hands, ankles and feet are the lowest. This is not only due to the model being unstable in these regions, but also due to our generated ground truth being noisy in these regions, since SMPL does not receive information about foot or hand orientation during the



(a) Segmentation. **(b)** Landmark detection.

Figure 8.4: *Visualization of the discriminative model scores.*

fits other than the foreground segmentation. The mean IoU score over all parts is 0.4560, and the accuracy 0.9132. The keypoint score visualization shows the same pattern, with the lowest scores at the big toes. The overall stability is very high with an average PCK@0.2 of 0.9450 (for the 14 core keypoints: 0.9388). You can find additional qualitative results in Figure E.1.

8.3.5 Direct 3D Pose and Shape Prediction

The 91 landmark predictions in 2D enable a human observer to easily infer the 3D shape and pose of a person: the keypoints on the body surface provide a good hint to estimate person shape and in combination with the skeleton orientation and pose can usually be identified (*c.f.* Figure 8.3b). This observation inspired us to explore the limits of a predictor for the 3D body model parameters directly from the 2D keypoint input. For this purpose, we use the 3D poses and shapes from the UP-3D dataset to sample projected landmarks with the full 3D parameterization of SMPL as labels. We move a virtual 'camera' for every pose on 5 elevations to 36 positions around the 3D model to enhance the training set.

On this data, we experimented with multi-layer perceptrons as well as Decision Forests. We preferred the latter regressor, since Decision Forests are less susceptible to noise. We train a separate forest to predict the axis-angle rotation vector for each of the 24 SMPL joints, as well as one to predict the depth. The input landmark

positions are normalized w.r.t. position and scale to improve generalization. We experimented with distance-based features and dot-product features from the main skeleton connections, but these were not as robust as plain 2D image coordinates. It turned out to be critical to use full rotation matrices as regression targets: the axis-angle representation has discontinuities, adding noise to the loss function.

One Decision Forest predicts pose or shape in ∼0.13s[4]. The predictions of all forests are independent, which means that the full pose and shape prediction can be obtained in between one and two orders of magnitudes faster than with SMPLify. This could allow the use of 3D pose and shape estimation for video applications, *e.g.*, action recognition.

Whereas the 3D model configuration does not always match the image evidence (see Figure 8.3d), it recovers the rough pose. We provide scores in Table 8.3 and Table 8.4 with the name 'DP' (**D**irect **P**rediction). We additionally add the scores for a hybrid version, for which we predict pose and shape using Decision Forests and take few optimization steps to make the global rotation of the body model match the image evidence (with varying runtime depending on the initialization, but less than one second on our data).

The difference between the full optimization on the LSP dataset in f1 score is 0.1062 for the 91 landmark based method and reduces with rotation optimization to 0.086. On the 3D datasets, the direct prediction method outperforms all optimization based methods except for SMPLify that runs in the order of tens of seconds.

Together with our ResNet101-based CNN model, it is possible to predict a full 3D body model configuration from an image in 0.378s. The pose-predicting CNN is the computational bottleneck. Because our findings are not specific to a CNN model, we believe that by using a speed-optimized CNN, such as SqueezeNet [IMA+16], and further optimizations of the direct predictor, the proposed method could reach real-time speed.

8.4 Closing the Loop

With the improved results for 3D fitting, which helped to create the dataset of 3D body model fits in the first place, a natural question is, whether the improved fitting method helps to enlarge the dataset.

We ran SMPLify on the 91 landmark predictions from our pose estimator and again asked human annotators to rate the 3D fits on all LSP images that were not accepted for our initial dataset. Of the formerly unused 54.75% of the data (1095 images), we found an improvement in six body part segmentation f1 score for 308 images (*c.f.* Figure 8.5a). We show three example images with high improvement

[4]For all timings, a test system with a 3.2Ghz six core processor and an NVIDIA GeForce GTX970 has been used.

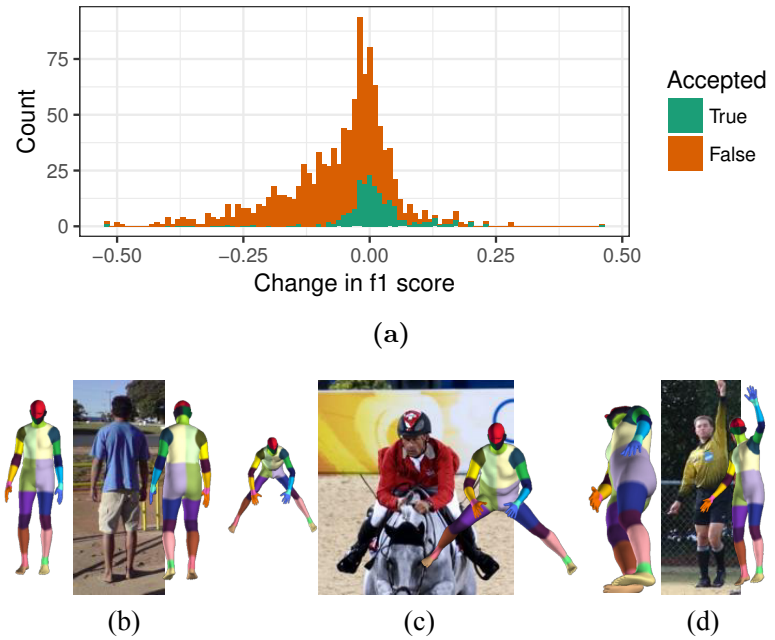


Figure 8.5: *Improvement of body model fits on the unused part of the* LSP *dataset.* Compared are fits to 91 predicted keypoints vs. fits to 14 ground truth keypoints. **(a)**: histogram of change in f1 score of projected six body part segmentation agreement with human annotator ground truth. Green color indicates that the formerly unaccepted fit to the 14 ground truth keypoints is accepted as valid when performed with the 91 predicted keypoints. For each image triple in **(b)**, **(c)**, **(d)**: left: SMPLify fit to the 14 ground truth keypoints, right: fit to the predicted 91 landmarks from our predictor. [LRK+17] © 2017 IEEE.

in f1 score in Figure 8.5, (b) to (d): improvement due to left-right label noise, depth ambiguity and perspective resolution compared to fits on the 14 ground truth keypoints. Further examples for improvements due to better perspective resolution and more detailed estimation of limb rotations can be found in Figure 8.6. Human annotators accepted additional 185 images, which is an improvement of 20% over the number of accepted initial fits and an absolute improvement of 9.3% in accepted fits of the LSP dataset.

The most common reasons for improvement are (1) noisy annotations, (2) better perspective resolution and (3) the better match of keypoints to the SMPL skeleton. An even higher ratio of improvement can be expected for the datasets with more annotation noise, such as the LSP-extended and MPII-HPDB datasets. This enlarged set of data could be used to again train estimators and continue iteratively.

Additional examples of improved fits from using 91 keypoints from our predictor over fits to the 14 ground truth keypoints can be found in Figure 8.6. We only show one fit that improves due to wrong ground truth labels (first row, rightmost triple), even though this is a frequent source of improvement. Instead, we highlight that

Figure 8.6: *Improvements due to better perspective resolution and limb rotation estimation.* For each image triple: improvement over fits to 14 ground truth keypoints (left) by using 91 keypoints from our predictor (center, right) on the LSP dataset.

the fits take more details into account and the additional keypoints provide hints to disambiguate perspective and rotation. With the cues about limb rotation, the fits look more realistic. We expect this to improve the pose estimator even further once the additional samples are integrated into the dataset.

8.5 Discussion

With the presented method and dataset, we argue for a holistic view on human related prediction tasks. By improving the representation of humans we could integrate datasets with different annotations and approach established tasks at a new level of detail. The presented results include high fidelity semantic body part segmentation into 31 parts and 91 landmark human pose estimation. This sets a new mark for estimating details of human body pose and shape that previous work did not reach. At the same time, it helps to improve the state-of-the art for 3D human pose estimation on the two standard benchmark datasets HumanEva and Human3.6M.

Furthermore, we present a regression tree model that predicts the 3D body configuration from 2D keypoints directly. This method runs orders of magnitude faster than optimization based methods. This direct prediction captures the overall pose from simple 2D input reasonably well and we are optimistic that it can be optimized to reach near real-time performance. The improved 3D fitting method produces more good fits that enlarge and refine the training set. Here, we only performed one iteration of the suggested iterative process but are confident that a system that iterates further can be deployed on large scale to continuously learn and improve with very limited human feedback.

Chapter 9

A Generative Model of People in Clothing

In the preceding chapters, we described how 2D and 3D models can be combined to create increasingly detailed descriptions of people. In this chapter, we address the inverse problem of automatically *generating* images of people in clothing. A traditional approach to this task is by means of computer graphics. A pipeline including 3D avatar generation, 2D pattern design, physical simulation to drape the cloth, and texture mapping is necessary to render an image from a 3D scene.

On the one hand, graphic pipelines provide precise control of the outcome. On the other hand, the rendering process poses various challenges, all of which are active research topics and mostly require human input. Especially clothing models require expert knowledge and are laborious to construct: the physical parameters of the cloth must be known in order to achieve a realistic result. In addition, modeling the complex interactions between the body and clothing and between different layers of clothing presents challenges for many current systems. The overall cost and complexity limits the applications of realistic cloth simulation. Data driven models of cloth can make the problem easier, but available data of clothed people in 3D is scarce.

Figure 9.1: *Random examples of people generated with our model.* For each row, sampling is conditioned on the silhouette displayed at the left. Our proposed framework also supports unconditioned sampling as well as conditioning on local appearance cues, such as color. [LPMG17a] © 2017 IEEE.

Here, we investigate a different approach and aim to side-step this pipeline. We propose ClothNet, a generative model of people learned directly from images. ClothNet uses task specific information in the form of a 3D body model, but is mostly data-driven. A basic version (*ClothNet-full*) allows to randomly generate images of people from a learned latent space. To provide more control we also introduce a conditional model (*ClothNet-body*). Given a synthetic image silhouette of a projected 3D body model, ClothNet-body produces random people with similar pose and shape in different clothing styles (see Figure 9.1).

Learning a direct image based model has several advantages: firstly, we can leverage large photo-collections of people in clothing to learn the statistics of how clothing maps to the body; secondly, the model allows to dress people fully automatically, producing plausible results. Finally, the model learns to add realistic clothing accessories such as bags, sunglasses or scarfs based on image statistics.

We run multiple experiments to assess the performance of the proposed models. Since it is inherently hard to evaluate metrics on generative models, we show representative results throughout the paper and explore the encoded space in a principled way in several experiments. To provide an estimate on the perceived quality of the generated images, we conducted a user study. With a rate of 24.7% or more, depending on the ClothNet variant, humans take the generated images for real.

9.1 Related Work

9.1.1 3D Models of People in Clothing

There exists a large and diverse literature on the topic of creating realistic looking images of people. They can be grouped into rendering systems and systems that attempt to modify existing real photographs (warping pixels).

Figure 9.2: *Sample results for virtual humans from existing approaches (ltr.):* [PJW+11], [PJA+12], local warping. [IPOS14], [VRM+17], animated 3D scans in real environments. [VLM+14], 3D avatars in virtual environments. [CWL+16], 3D avatars in real environments. See the text for a more detailed discussion. [LPMG17a] © 2017 IEEE.

Warping pixels. Xu et al. [XLS+11] pose the problem as one of video retrieval and warping. Rather than synthesizing meshes with wrinkles, they look up video frames with the right motions. Similarly, in [ZFL+10, JTST10] an unclothed body model is fit to multi-camera and monocular image data. The body is warped and the image reshaped accordingly.

Two prominent works that aim to reshape people in photos are [PJW+11, PJA+12] (*c.f.* Figure 9.2). A number of different synthetic sources has been used in [PJW+11] for improvement of pedestrian detection. The best performing source were morphed images of people, but requires data from a multi-view camera setup; consequently only 11 subjects were used. Subsequent work [PJA+12] reshaped images but required significant user interaction. All aforementioned approaches require manual input and can only modify existing photographs.

Rendering systems. Early works synthesizing people from a body model are [SSK+13, TSSF12, PMTS+13]. Here, renderings were limited to depth images with the goal of improving human pose estimation from depth data. The work of [CWL+16] combines real photographs of clothing with a SCAPE [ASK+05] body model to generate synthetic people whose pose can be controlled (*c.f.* Figure 9.2). For the same task, the work of [RS16] proposes a pose-aware image blending technique, the final images are composed of blends of multiple images from a set of 2D images, limiting the ability to generalize.

A different line of works use rendering engines with different sources of inputs. In [IPOS14], a mixed reality scenario is created by rendering 3D rigged animation models into videos, but it is clearly visible that results are synthetic (*c.f.* Figure 9.2). The work of [VRM+17] combines a physical rendering engine together with real captured textures and motion capture data to generate novel views of people. All these works use 3D body models without clothing geometry, hence the quality of the results is limited. One exception is [GFB10] where only the 2D contour of the projected cloth is modeled.

3D clothing models. Much of the recent work in the field of clothing modeling is focused on how to make simulation more computationally efficient [GHF+07, NSO12], particularly by adding realistic wrinkles to low-resolution simulations [KGBS11, KKN+13]. Other approaches have focused on taking off-line simulations and learning data driven models from them [dASTH10, SGdA+10, GRH+12, KKN+13, WOR11]. The authors of [RKS+14] move a 3D body model to a monocular video sequence, simulate clothing on the body, and then backproject the simulated garment. All these approaches require predesigned garment models. Furthermore, current 3D models are not fully automatic, restricted to a set of garments or not photorealistic.

9.1.2 Generative Models

Variational models and GANs. Variational methods are a well-principled approach to build generative models. Kingma and Welling developed the Variational Autoencoder [KW13], which is a key component of our method. In their original work, they experimented with a multi-layer perceptron on low resolution data. Since then, multiple projects have designed Variational AutoEncoders (VAEs) for higher resolutions, *e.g.*, [YYSL16]. They use a Conditional Variational AutoEncoder (CVAE) [KMRW14, SLY15] to condition generated images on vector embeddings. Recurrent formulations [GDG+15, vdOKK16, vdOKE+16] enable to model complex structures, but again only at low resolution. With [DCSF15], Denton et al. address the resolution issue explicitly and propose a general architecture that can be used to improve network output. This strategy could be used to enhance ClothNet. Generative Adversarial Networks [GPAM+14] use a second network during the training as an adversary, distinguishing between training data and generated data, to enhance the loss function of the trained network. We use this strategy in our training to increase the level of detail of the generated images. Most of the discussed works use resolutions up to 64x64 while we aim to generate 256x256 images. For our model design we took inspiration from encoder-decoder architectures such as the U-Net [RFB15], Context Encoders [PKD+16] and the image-to-image translation networks [IZZE16].

Inpainting methods. Recent inpainting methods achieve a considerable level of detail [PKD+16, YLL+16] in resolution and texture. To present a comparison with a state-of-the art encoder-decoder architecture, we provide a comparison with [PKD+16] in our experiments. [SWH+16] works directly on a texture map of a 3D model. Future work could explore to combine it with our approach from 2D image databases.

Deep networks for learning about 3D objects. There are several approaches to reason about 3D object configuration with deep neural networks. The work of Kulkarni [KWKT15] use VAEs to model 3D objects with very limited resolution and assume that a graphics engine and object model are available at learning time. In [DQN16] an encoder-decoder CNN in voxel space is used for 3D shape completion, which requires 3D ground truth. The authors of [OWL15] develop a generative model to create depth training data for articulated hands. This avoids the problem of generating realistic appearance.

9.2 Chictopia made SMPL

To train a supervised model connecting body parameters to fashion, we need a dataset providing information about both. Datasets for training of pose estimation

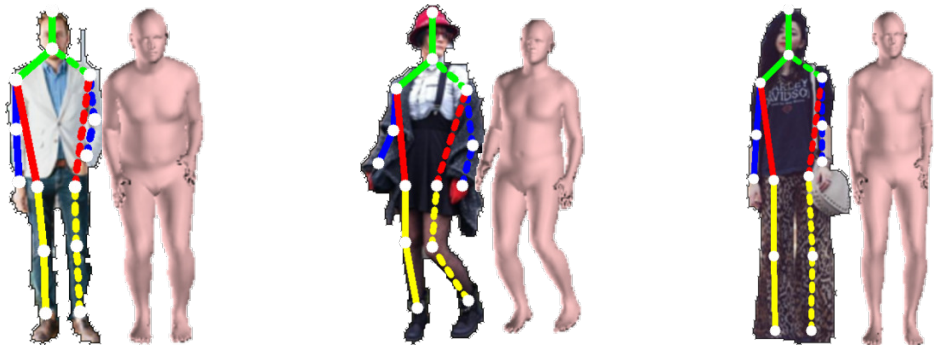


Figure 9.3: *Example images from the Chictopia10K dataset [LXS+15], detected joints from the DeeperCut CNN [IPA+16] and the SMPLify fits.* Typical failure cases are foot and head orientation (**center**). The pose estimator works reliably even with wide clothing and accessories. (**right**). [LPMG17a] © 2017 IEEE.

systems [APGS14, JE10, IPOS14] as well as the UP-3D dataset introduced in Chapter 8 capture complex appearance, shape and pose variation, but are not labeled with clothing information. The Chictopia10K dataset [LXS+15] contains fine-grained fashion labels but no human pose and shape annotations. In the following sections, we explain how we automatically augmented Chictopia10K so that it can be used as a resource for generative model training.

9.2.1 Fitting SMPL to Chictopia10K

The *Chictopia10K* dataset consists of 17,706 images collected from the chictopia fashion blog[1]. For all images, a fine-grained segmentation into 18 different classes (*c.f.* [LXS+15]) is provided: 12 clothing categories, background and 5 features such as hair and skin (see Figure 9.4). For shoes, arms and legs, person centric left and right information is available. We augment the *Chictopia10K* dataset with pose and shape information by fitting the 3D SMPL body model [LMR+15] to the images using the SMPLify pipeline (see Chapter 6). SMPLify requires a set of 2D keypoint locations which are computed using the DeeperCut CNN [IPA+16] pose estimator. We selected the DeeperCut CNN out of convenience, any other recent keypoint detector [WRKS16, NYD16] will likely produce similar results. We use the SMPLify energy optimization from Equation 6.1 on the estimated joints to obtain 3D fits. Qualitative results of the fitting procedure are shown in Figure 9.3. The pose estimator has a high performance across the dataset and the 3D fitting produces few mistakes. The most prominent failures are results with wrong head and foot orientation. To leverage as much data as possible to train our supervised models, we refrain from manually curating the results. Since we are interested in overall body shape and pose, we are using a six-part segmented projection of the SMPL fits for conditioning of ClothNet-body. Due to the rough segmentation, segmented areas are still representative even if orientation details do not match.

[1] `http://www.chictopia.com/`

Figure 9.4: *Example annotations from the Chictopia10K dataset [LXS+15] before and after processing* (for each pair **left** and **right** respectively). Holes are inpainted and a face shape matcher is used to add facial features. The rightmost example shows a failure case of the face shape matcher. [LPMG17a] © 2017 IEEE.

9.2.2 Face Shape Matching and Mask Improvement

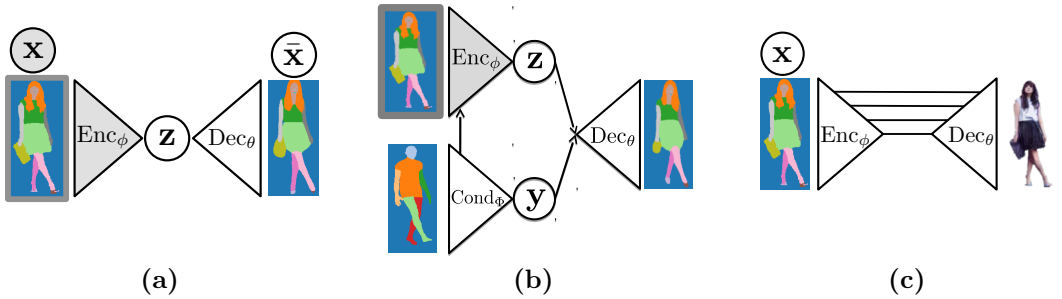
We further enhance the annotation information of Chictopia10K and include facial landmarks to add additional guidance to the generative process. With only a single label for the entire face, we found that all models generate an almost blank skin area in the face.

We use the dlib [Kin09] implementation of the fast facial shape matcher [KS14] to enhance the annotations with face information. We reduce the detection threshold to oversample and use the face with the highest IoU score of the detection bounding box with ground truth face pixels. We only keep images where either no face pixels are present or the IoU score is above a certain threshold. A threshold score of 0.3 was sufficient to sort out most unusable fits and still retain a dataset of 14,411 images (81,39%).

Furthermore, we found spurious "holes" in the segmentation masks to be problematic for the training of generative models. Therefore, we apply the morphological "close" and "blackhat" operations to fill the erroneously placed background regions. We carefully selected the kernel size and found that a size of 7 pixels fixes most mistakes while retaining small structures. You can find examples of original and processed annotations in Figure 9.4.

9.3 ClothNet

Learning a generative model of images of dressed people is challenging. Currently, the visually most appealing technique to create fine-grained textures at 256x256 resolution are image-to-image translation networks [IZZE16]. They rely on a well designed encoder-decoder structure, with skip connections between correspondingly shaped layers of encoder and decoder. This allows the model to retain sharp edges faithful to its input.



(a) **(b)** **(c)**

Figure 9.5: *ClothNet modules*: **(a)** the latent sketch module consists of a variational auto-encoder, **(b)** the conditional sketch module consists of a conditional variational auto-encoder and **(c)** the portray module is an image-to-image translation network that fills a sketch with texture. The modules in (a) and (c) are concatenated to form ClothNet-full and modules (b) and (c) are concatenated to form ClothNet-body. The learned latent representation $\mathbf{z}$ in (a) and (b) is a 512-D random variable that follows a multivariate Gaussian distribution. The variable $\mathbf{y}$ is a deterministic latent encoding of the body model silhouette that we use to condition on pose and shape. At test time in (a) and (b) one can generate a sample from the multivariate Gaussian $\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and push them through the decoder network to produce random sketches of people in different clothing. We show in (a) and (b) the input to the encoder in gray color, this indicates they are not available at test time. [LPMG17a] © 2017 IEEE.

However, using image-to-image translation networks directly for the task of predicting dressed people from SMPL sketches as displayed in Figure 9.1 does not produce good results (we provide example results from such a model in Figure 9.10b). The reason is that there are many highly different completions possible for a single 3D pose. The image-to-image translation model has not the required structures to handle such situations. Furthermore, its sampling capabilities are poor. Variational Autoencoders, on the other hand, excel at encoding high variance for similar inputs and provide a principled way of sampling.

We combine the strengths of both, Variational Autoencoders and the image-to-image translation models, by stacking them in a two-part model: the *sketch* part is variational and deals with the high variation in localized image appearance. Its output is a semantic segmentation map (sketch) of a dressed person. The second *portray* part uses the created sketch to generate an image of the person and can make use of skip connections to produce sharp and detailed output. The intermediate representation as a sketch enables us to experiment with different modules for the two model parts. In the following sections, we introduce the modules we experimented with.

9.3.1 The Latent Sketch Module

The latent sketch module is a variational auto-encoder which allows to sample random sketches of people consisting of their clothing segments.

The Variational Auto-Encoder [KW13] consists of two parts, an *encoder* to a latent space, and a *decoder* from the latent space to the original representation. As for any latent variable model, the aim is to reconstruct the training set $\boldsymbol{x}$ from a latent representation $\boldsymbol{z}$. Mathematically, this means maximizing the data likelihood $p(\boldsymbol{x}) = \int p_\theta(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z}$. In high dimensional spaces, finding the decoder parameters θ that maximize the likelihood is intractable. However, for many values of $\boldsymbol{z}$ the probability $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ will be almost zero. This can be exploited by finding a function $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, the *encoder*, parameterized by ϕ. It encodes a sample $\boldsymbol{x}^i$ and produces a distribution over $\boldsymbol{z}$ values that are likely to reproduce $\boldsymbol{x}^i$. To make the problem tractable and differentiable, this distribution is assumed to be Gaussian, $q_\phi(\boldsymbol{z}|\boldsymbol{x}^i) = \mathcal{N}(\boldsymbol{\mu}^i, \boldsymbol{\Sigma}^i)$. The parameters $\boldsymbol{\mu}^i, \boldsymbol{\Sigma}^i$ are predicted by the ϕ-parameterized encoding neural network Enc_ϕ. The decoder is the θ parameterized neural network Dec_θ.

Another key assumption for VAEs is that the marginal distribution on the latent space is Gaussian distributed with zero mean and identity covariance, $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Under these assumptions, the VAE objective (see [KW13] for derivations) to be maximized is

$$\sum_i \boldsymbol{E}_{z\sim q}[\log p_\theta(\boldsymbol{x}^i|\boldsymbol{z})] - D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x}^i)||p(\boldsymbol{z})), \tag{9.1}$$

where $\boldsymbol{E}_{z\sim q}$ indicates expectation over distribution q and D_{KL} denotes Kullback-Leibler (KL) divergence. The first term measures the decoder accuracy for the distribution produced by the encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ and the second term penalizes deviations of $q_\phi(\boldsymbol{z}|\boldsymbol{x}^i)$ from the desired distribution $p(\boldsymbol{z})$. Intuitively, the second term prevents the encoding from carrying too much information about the input $\boldsymbol{x}$. Since both $q_\phi(\boldsymbol{z}|\boldsymbol{x}^i)$ and $p(\boldsymbol{z})$ are Gaussian, the KL divergence can be computed in closed form [KW13]. Equation 9.1 is maximized using stochastic gradient ascent.

Computing Equation 9.1 involves sampling; constructing a sampling layer in the network would result in a non differentiable operation. This can be circumvented using the reparameterization trick [KW13]. With this adaptation, the model is deterministic and differentiable with respect to the network parameters θ, ϕ. The latent space distribution is forced to follow a Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ during training. This implies that at test time one can easily generate samples $\bar{\boldsymbol{x}}^i$ by generating a latent sample $\boldsymbol{z}^i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and pushing it through the decoder $\bar{\boldsymbol{x}}^i = \text{Dec}_\theta(\boldsymbol{z}^i)$. This effectively ignores the encoder at test time.

Sketch encoding: we want to encode images $\boldsymbol{x} \in \mathbb{R}^{256\times256}$ of sketches of people into a 512-D latent space, $\boldsymbol{z} \in \mathbb{R}^{512}$. This resolution requires a sophisticated encoder and decoder layout. Hence, we combine the recently proposed encoder and decoder architecture for image-to-image translation networks [IZZE16] with the formulation of the VAE. We use a generalized Bernoulli distribution to model $p_\theta(\boldsymbol{x}^i|\boldsymbol{z})$. The architecture is illustrated schematically in Figure 9.5(a).

9.3.2 The Conditional Sketch Module

For some applications it may be desirable to generate different people in different clothing in a pose and shape specified by the user. To that end, we propose a module that we call conditional sketch module.

The conditional sketch module gives control of pose and shape by conditioning on a 3D body model sketch as illustrated in Figure 9.5(b). We use a conditional variational autoencoder for this model (for a full derivation and description of the idea, we refer to [KMRW14]). To condition on an image $\boldsymbol{Y} \in \mathbb{R}^{256\times256}$ (a six part body model silhouette), the model is extended with a new encoding network Cond_Φ, with similar structure as Enc_ϕ. Since the conditioning variable is deterministic, the encoding is $\boldsymbol{y} = \text{Cond}_\Phi(\boldsymbol{Y})$. To provide the conditioning input to the encoder, we concatenate the output of the first layer of Cond_Φ to the output of the first layer of Enc_ϕ. To train the model, we use the same objective as in Equation 9.1. Here, the decoder reconstructs a sample using both, $\boldsymbol{z}$ and $\boldsymbol{y}$, with $\hat{\boldsymbol{x}}^i = \text{Dec}_\theta(\boldsymbol{y}^i, \boldsymbol{z}^i)$ and the minimization of the KL-divergence term is only applied to $\boldsymbol{z}$.

9.3.3 The portray Module

For applications requiring a full textured image of the person, the sketch modules can be chained to a portray module. To that end, we trained a conditional auto-encoder for texture as well. Here, we condition on a cloth sketch and sample to produce different textures and colors. Unfortunately, in our experiments the learned latent space is not producing variation on the output and the model operates almost as a deterministic predictor. We believe the model has not seen enough data to generalize; in particular there are not enough similar sketches that map to many different kinds of textures. Instead, we use the recently released image-to-image translation model [IZZE16] with minimal modifications to color the sketches produced by the latent sketch modules. With additional face information, we found this model to produce appealing results.

9.3.4 ClothNet-full and ClothNet-body

Once the sketch part and the portray part are trained, they can be concatenated to obtain a full generative model of images of dressed people. We refer to the

concatenation of the latent sketch module with the portray module as ClothNet-full. The concatenation of the conditional sketch module with the portray module is named ClothNet-body. Several results produced by ClothNet-body are illustrated in Figure 9.1. All stages of ClothNet-full and ClothNet-body are differentiable and implemented in the same framework. We trained the sketch and portray modules separately, simply because it is technically easier. Propagating gradients through the entire model is possible and may improve results, but we did not explore this direction yet.

9.3.5 Network Architectures

Adhering to the image-to-image translation network architecture for designing encoders and decoders, we make use of LReLUs [MHN13], batch normalization [IS15] and use fractionally strided convolutions [ZTF11]. Even for this highly parameterized architecture and using the adaptive ADAM optimizer [KB14], we found that the learned latent distribution is not forced close enough to a standard normal distribution, causing unrealistic results when sampling. Hence, we introduced weight parameters for the two loss components in Equation 9.1 and upweigh the KL component by factor 7. For a comprehensive overview of models and network designs, see Figure F.1.

9.4 Experiments

In a comprehensive series of experiments, we want to shed light on all aspects of the proposed model. Since an interplay of multiple components is required, we break the experiments down into multiple sections.

9.4.1 The Latent Sketch Module

We provide numbers on the quality of reconstruction in Table 9.1. The values are means of the respective metrics over all classes. The overall reconstruction accuracy

Figure 9.6: *A walk in latent space along the dimension with the highest variance.* We built a PCA space on the 512 dimensional latent predictions of the test set and walk -1STD to 1STD in equidistant steps. [LPMG17a] © 2017 IEEE.

Model	Part	Accuracy	Precision	Recall	F1
VAE	Train	0.958	0.589	0.584	0.576
	Test	0.952	0.540	0.559	0.510
CVAE	Train	0.962	0.593	0.591	0.587
	Test	0.950	0.501	0.502	0.488

Table 9.1: *Reconstruction metrics for the latent and conditional sketch modules.* The overall reconstruction accuracy is high. The other metrics are dominated by classes with few labels. The CVAE overfits faster. [LPMG17a] © 2017 IEEE.

is high with an accuracy score of more than 0.95 in all settings. The other metrics are influenced by the small parts, in particular facial features. The CVAE overfits faster than the VAE, due to the less regularized information from the conditioning.

For a generative model, qualitative assessment is important as well. For this, we provide a visualization of a high variance dimension in latent space in Figure 9.6. To create it, we generated samples $\boldsymbol{z}$ from all test set images. To linearize their representation, we used the Cumulative Distribution Function (CDF) on the values. We then used a PCA to discover the direction of most variance. In the PCA space, we take evenly spaced steps from one standard deviation in negative to positive direction, *i.e.*, the PCA mean image is in the center of Figure 9.6.

Even though the most dominant direction in PCA space only encodes roughly 1% of the variance, the complexity of the task becomes obvious: even the first dimension encodes variations in pose, shape, position, scale and clothing types. The model learns to adjust the face direction in plausible ways.

9.4.2 The Conditional Sketch Module

As described in Section 9.3.2, we use a CVAE architecture to condition the generated clothing segmentations. We use the SMPL body model to represent the conditioning. However, instead of using the internal SMPL representation as vector of shape components and angle rotations, we instead render the SMPL body in the desired configuration. We use six body parts: head, central body, left and right arms, left and right legs, to give the model local cues about the body parts.

We found the six part representation to be a good trade-off: using only a foreground-background encoding may convey too little information, especially about left and right parts. A too detailed segmentation introduces too much noise, since the data for training our supervised models has been acquired by automatic fits solely to keypoints. These fits may not represent detailed matches in all cases. You can find qualitative examples of conditional sampling in Figure 9.1 and Figure 9.7.

At test time, we encode the model sketch (Figure 9.7(a)) to obtain $\boldsymbol{y} = \mathrm{Cond}_\Phi(\boldsymbol{Y})$, sample from the latent space $\boldsymbol{z}^i \sim p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and obtain a clothed sketch with $\bar{\boldsymbol{x}}^i = \mathrm{Dec}_\theta(\boldsymbol{y}, \boldsymbol{z}^i)$. For every sample $\boldsymbol{z}^i$ a new sketch $\hat{\boldsymbol{x}}^i$ is generated with different

(a) (b) (c)

Figure 9.7: *Conditional sampling using ClothNet-body* Per row: **(a)** SMPL conditioning for pose and shape, **(b)** sampled dressed sketches conditioned on the same sample in (a), **(c)** the nearest neighbor of the rightmost sample in (b) from the training set. The model learns to add various hair types, style and accessories. [LPMG17a] © 2017 IEEE.

clothing but roughly the same pose and shape. Notice how different samples produce different hair and cloth styles as well as different configurations of accessories such as bags.

9.4.3 Conditioning on Color

As an example for adding additional conditioning, we describe how to condition our model on color. For every image in the training set, we create a new image by coloring the sketch parts with their respective median color. The concatenation of the per-part median colored image and the sketch are the inputs to a new portray module which is trained on this input. Conditioning can then be achieved by manually selecting a color for each sketch segment. Example results are shown in Figure 9.8. The network learns to follow the color cues, but still does not only generate plain color clothing, but places patterns and textures.

9.4.4 ClothNet

With the following two experiments, we want to provide an insight in how realistic the images are that are generated from the ClothNet-full pipeline.

Figure 9.8: *Conditioning on color*: (**left**) sketch input to the network. (**right**) Four different outputs for four different color combinations. Color conditioning for the regions are shown in the boxes below the samples (**boxes below, ltr**): lower dress, upper dress, jacket, hat. [LPMG17a] © 2017 IEEE.

Figure 9.9: *Results from ClothNet with added random backgrounds.* **First row**: results from ClothNet-full (*i.e.*, sketch and texture generation), **second row**: results from the portray module and ground truth sketches. [LPMG17a] © 2017 IEEE.

9.4.4.1 Generating an Artificial Dataset

In the first experiment, we generate an artificial dataset and train a semantic segmentation network on the generated data. By comparing the performance of a discriminative model trained on real or synthetic images we can asses how realistic the generated images are.

For this purpose, we generate an equally sized dataset to our enhanced subset of Chictopia10K. We store the semantic segmentation masks generated from the latent sketch module as artificial 'ground truth' and the outputs from the full ClothNet-full pipeline as images. To make the images comparable to the Chictopia10K images, we add artificial background. Similar to [VRM+17], we sample images from the *dining room*, *bedroom*, *bathroom* and *kitchen* categories of the LSUN dataset [YZS+15]. Example images are shown in Figure 9.9.

Train \ Test	Full Synth.	Synth. Text.	Real
Full Synth.	0.566 0.978	0.437 0.964	0.335 0.898
Synth. Text.	0.503 0.968	0.535 0.976	0.411 0.915
Real	0.448 0.955	0.417 0.957	0.522 0.951

Table 9.2: *Segmentation results* (per line: IoU, accuracy) for a variety of training and testing datasets. **Full Synth.** results are from the ClothNet-full model, **Synth. Text.** from the portray module on ground truth sketches. [LPMG17a] © 2017 IEEE.

Model	Real image rated gen.	Gen. image rated real
ClothNet-full	0.154	0.247
portray mod.	0.221	0.413

Table 9.3: *User study results from 12 participants.* The first row shows results for the full ClothNet-full model, the second for the portray module used on ground truth sketches. [LPMG17a] © 2017 IEEE.

Even though the generated segmentations from our VAE model look mostly realistic for a human observer, some weaknesses become apparent when completed by the portray module: bulky arms and legs and overly smooth outlines of fine structures such as hair. Furthermore, the different statistics of facial landmark size to ground truth sketches lead to less realistic faces.

We train a DeepLab ResNet-101 [CPK+15] segmentation model on real and synthetic data and evaluate on test images from all data sources and provide the results in Table 9.2. As expected, the models trained and tested from the same data source perform best. The model trained on the real dataset reaches the highest performance and can be trained longest without overfitting. The fully synthetic datasets, however, do not lose that much accuracy compared to the model trained on real data. The IoU scores, however, suffer from fewer fine structures present in the generated data that dominate the scores. We experimented with a combined dataset consisting of real and artificial data from the portray module and found an improvement compared to a model trained solely on real data on the edge of significance.

9.4.4.2 User Study

We performed a user study to quantify the realism of images for humans. We set up an experiment to evaluate both stages of our model: one for images generated from the portray module on ground truth sketches and once for the full ClothNet-full model.

(a) (b)

Figure 9.10: *Example results* from **(a)** the context encoder architecture [PKD+16] from a ground truth sketch. Without skip connections, the level of predicted detail remains low. **(b)** Results from an image-to-image network trained to predict dressed people from six part SMPL sketches directly. Without the proposed two-stage architecture, the model is not able to determine shape and cloth boundaries. [LPMG17a] © 2017 IEEE.

For each of the experiments, we ask users to label 150 images on whether they believe are a photo or generated from our model. 75 images are real Chictopia images, 75 generated with our model. Every image is presented for 1 second akin to the user study in Isola et al. [IZZE16]. They use a forced choice between two images, one ground truth, one sketched by their model from ground truth segmentation.

However, since we do not have ground truth comparison images, we refrain to display one image at a time for a forced choice. This setting is slightly harder for our model, since the user can focus on one image. To assess overall image quality and remove the human focus on faces, we blank out the faces for artificial and real images equally. The first 10 images are ignored in the evaluation to let users calibrate on image quality. The results for 12 participants are presented in Table 9.3. Even for the fully generative pipeline, users are fooled 24.7% of the time. For comparison: Isola et al. [IZZE16] report fooling rates of 18.9% and 6.1%, however on other modalities.

9.4.4.3 Additional Qualitative Results

The results shown in Figure 9.11 conclude and summarize our model evaluation. The first set in Figure 9.11a consists of additional results from ClothNet-full without background completion (*c.f.* Figure 9.9). The model generates plausible people in a diverse set of poses and with a variety of clothes. Larger disconnected components hardly occur, an example can be found in the second row, rightmost picture. Failure cases can observed where the model produces unrealistic body proportions.

(a)

(b)

(c)

Figure 9.11: *Results from the proposed models.* **(a)** Results from ClothNet-full. **(b)** Results from ClothNet-body. Crossed legs in the third row are sometimes being rendered as crossed or near parallel. This is due to label noise in the training data for the *conditional sketch module* (see Section 9.4.4.3 for a full discussion). **(c)** Results from the *portrait* module applied to ground truth sketches from the test set.

In Figure 9.11b, we present additional results from ClothNet-body. The first two rows show textured samples from conditioning on the poses presented in Figure 9.10b. Conditioned on a fixed pose and body shape, ClothNet-body generates a variety of clothing types and textures. The third row shows results for a cross-legged pose. Here, the *conditional sketch module* generates both, cross-legged and straight-legged samples with the legs close together. This is due to 3D fitting noise of the body model to the Chictopia10K dataset: cross-legged fits are often erroneous fits to people with straight legs close together and vice versa. The model learns to reproduce this variety observed in the training data. Improved 3D fits would resolve this problem.

In Figure 9.11c, we show results for the *portrait* module applied on ground truth sketches from the test set. Most results are plausible, with detailed wrinkles and hair. The model occasionally uses colorful patterns on dresses and tops.

9.5 Discussion

In this chapter, we developed and analyzed a new approach to generate people with accurate appearance. We find that modern machine learning approaches may be powerful enough to sidestep traditional graphics pipeline design and 3D data acquisition. This is a first step and we anticipate that the results will become better once more data is available for training.

We enhanced the existing Chictopia10K dataset with face annotations and 3D body model fits. With a two-stage model for semantic segmentation prediction in the first, and texture prediction in the second stage, we presented a novel, modular take on generative models of structured, high-resolution images.

With this possibility to generate large amounts of training data at a very low computational and infrastructural cost together with the possibility to condition generated images on pose, shape or color, we see many potential applications for the presented method.

Part IV

Conclusion

Chapter 10

Summary and Discussion

Data-driven models provide an elegant way to cope with the complexity of human appearance. In this thesis, we (1) presented methods and frameworks to build and train such models and (2) showed how 2D and 3D, discriminative and generative models can be combined to parse and generate human appearance.

We first touched the topic of Decision Forest training, introduced and analyzed a new entropy function and established the relations to existing entropy functions and heuristics. The new entropy only respects two of the four Khinchin-Shannon axioms, but can be evaluated efficiently and tuned for a specific dataset. We implemented and ran the experiments with a new, object oriented Decision Forest library, which we made available as open source software. Thanks to its fine grained parallelization mechanisms, it can leverage modern manycore architectures even for node optimization and parallelize beyond tree level.

We introduced a second software package, *barrista*, an efficient and convenient interface for the popular deep learning framework *caffe*. It supports the easy design, training and execution of ANNs and CNNs. As opposed to plain *caffe*, it is not dependent on (but fully supports) protobuf network specifications. In particular, it makes designing networks possible from Python, even procedurally. This is an important advantage for very deep or multi column networks such as ResNets [HZRS16b] or multi-resolution networks, *e.g.*, [CPK+15].

In the following chapters, we put the hitherto developed frameworks to work for 2D and 3D modeling of humans. Our starting point were 2D images and we described methods to get 3D model fits from them, established a virtuous cycle to improve the fitting process and developed a generative model vice versa: generating realistic 2D images of humans from 3D body information.

We first described a method to automatically fit the 3D SMPL body model to a 2D image of a person. We used *barrista* to run a pose estimation CNN, producing 14 landmark estimates for the main human skeletal joints. We showed that a gradient based energy optimization with carefully designed prior terms is enough to optimize for 3D pose and body shape based on these 14 2D positions. Furthermore, we analyzed the interplay of detected keypoints, estimated segmentation and 3D fits in Chapter 7. We managed to improve segmentation by incorporating the keypoint

information and described a method to use segmentation information to improve trust assessment for estimated keypoints. Finally, we described how estimated silhouettes can be incorporated into the gradient based 3D fitting method. These techniques enabled us to automatically extract 3D body representations from 2D photos.

In Chapter 8, we established a virtuous cycle between fitting 3D models and using their projections to improve the 2D predictors providing the information for 3D fitting. This improved the granularity and the quality of the information used in the fitting process. Annotating the detailed body structures manually at the required level would be infeasible. Instead, we established a virtuous cycle between training 2D appearance models, 3D fitting and *curating* 3D fits (this was the only required human annotator involvement), projecting 2D labels and iterating. With this strategy, we trained 2D appearance models with five times more keypoints/segments than in the literature so far, which in turn helped us to push the state-of-the art in 3D human pose and body shape estimation.

Finally, we derived a generative model for people in clothing in Chapter 9. It is the first model that allows conditioned or unconditioned sampling of dressed people in a full body setting. The model is composed of two stages with interchangeable parts and is very flexible in its application. Further conditioning (apart from body pose and shape) can easily be integrated; we showcased this with the specification of desired cloth color.

With the presented techniques, we covered a full cycle of mapping human appearance 2D→3D→2D and made contributions to several aspects for systems that automatically understand and generate human appearance: learning techniques, implementation and modeling. The presented software libraries are already being used in research projects and the techniques for automatic fitting of a 3D body model and for fine-grained body pose and shape analysis are already implemented in products[1]. Developing these techniques further and making accurate 3D human body shape and pose estimation work in real-time on mobile platforms is an appealing aim for artificial intelligence. It will open up possibilities for interactive machine learning by enabling new annotation or—generally speaking—teaching paradigms: everyday users will be able to 'correct' their computer vision enabled systems by pointing at their environment and providing additional information. Such very natural interaction possibilities will provide a new and powerful source of data.

The presented methods present a step towards this aim, but further improvements will have to be made, mainly in terms of prediction quality and speed of execution. We continue with a discussion of limitations and criticism in the following section and conclude with an outlook on future work in Chapter 11.

[1] `https://www.bodylabs.com/soma/`

Figure 10.1: *The joint angle prior prefers slightly bent extremities.* In these examples, this is particularly visible for the legs, in the rightmost example also for the arms.

10.1 Limitations and Criticism

In this section, we discuss limitations and weaknesses of the presented approaches in the hope that they will serve as an inspiration for future work.

Decision Forest implementation. The fertilized forests library is written in C++ in an object oriented manner. All objects are, where applicable, templated with input, feature and output datatype as well as the result type. Whereas this guarantees high efficiency in terms of storage and at evaluation time, it also has several disadvantages: slow compilation due to limited application of the PIMPL idiom and unnecessary difficulties to derive and to control variable types and objects. We addressed both of them in the implementation (the first by isolating the especially compile time consuming serialization functions and the latter with factory objects and a code generator). The upcoming C++ 17 standard[2] comes with a new possibility to solve both problems in a significantly easier manner: `std::variant` objects can handle different datatypes with high efficiency. This elegantly removes the need for templated library objects completely, resulting in PIMPL implemented objects and an easy-to-use interface. We implemented a prototype demonstrating the potential of this technique at `http://github.com/classner/forpy`.

barrista was released in September 2015, short before google's Tensorflow framework was announced in November 2015[3]. This announcement motivated an increasing amount of people to migrate from existing frameworks to Tensorflow. This had a strong impact on the whole community, including the *caffe* and the *barrista* framework. Hence, we maintain the *barrista* framework, but will not develop it further. Instead, to ease scientific exchange, we implemented our latest presented model for generating people (Chapter 9) in Tensorflow. We're glad that some of the ideas we advertised with *barrista* were implemented in core *caffe*, *e.g.*, the 'NetSpecification' class for procedural model design.

[2] `http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4659.pdf`
[3] `https://www.wired.com/2015/11/google-open-sources-its-artificial-intelligence-engine/`

SMPLify is first and foremost limited by the appearance hints it receives. 14 keypoints at positions inside the body are insufficient to reliable disambiguate pose, shape and perspective. We address this issue in Chapter 8 with more fine-grained models. The method can be further improved w.r.t speed and initialization: gradient descent highly depends on the initialization and currently we are initializing with only one reasonable guess. Furthermore, the prior component of the energy function to prevent unrealistic joint angles $E_a(\boldsymbol{\theta})$ (Equation 6.3) is monotonous increasing. It is decreasing for increasingly bent extremities. Hence, we observe a tendency to prefer slightly bent extremities (see Figure 10.1). An additional step function could remove this effect. The information from the additional landmarks in Chapter 8 helps to dampen the effect.

A virtuous cycle for detailed 2D models and 3D fits. The *UP* dataset and models are currently limited in performance by missing information about head and foot orientation. We tolerated slight mismatches for these body parts when selecting the high quality fits, because they just can't be estimated from the currently available 2D cues. Being strict on matches for these body parts would reduce the dataset size to an unacceptable level. On the other hand, the models trained on projections from *UP-3D* seem to already pick up the slight correlation between toe and head appearance and 3D configuration. It remains an open and interesting question how many details the iterative process will pick up from the 3D fits performed only on 14 keypoints and when it saturates. Due to time constraints, we could not complete another iteration of the process. Another open challenge is to integrate hands, fingers and toes into the fitting and annotation process. Working with facial details and expressions probably requires a system that takes person identity into account.

Generative model. The main weakness of the presented generative model for people are the faces. We noticed a significant improvement from adding information about the facial landmarks to the ground truth sketches. The sketched facial landmarks are reproduced by the *portrait* module, but since humans are very sensitive to faces, even slight inaccuracies are noticed. A dedicated face model, *e.g.* [SWH+16], could solve this problem. A less often occurring problem are unnaturally round and wide sketches of extremities.

Chapter 11

Future work

The preceding section discusses potential areas of improvement from this work. In this section, we briefly describe a few concrete steps in more detail and conclude with a wide, more speculative outlook.

Towards a More Detailed, Large Scale Dataset of 3D People. With additional care and extensions, the UP datasets could evolve to a standard training and evaluation dataset for 3D human pose estimation from images in the wild. The following steps are important on the way: (1) improve head and foot pose estimation. To address the lacking details in rotation of head and feet, we already experiment with adding additional keypoints from human annotators. For this, we created an Amazon Mechanical Turk (AMT) labeling tool (see Figure 11.1). (2) Create a ground truth test set. Ideally, for this test set the camera parameters are known and 'real' 3D data is available to obtain high fidelity 3D fits. (3) Grow in size. To satisfy modern CNN architectures, ideally the training dataset would offer more data. The MS COCO keypoint dataset [LMB+14a] could provide a great start for an extended UP-3D version: foreground-background annotation is already available and the MS COCO database provides keypoints for another 185,316 images, including eye and ear positions. Hence, it could be easily integrated into the UP pipeline.

Towards mesh correspondance prediction from 2D. The already existing data of the UP-3D dataset allows us to investigate a whole range of questions concerning full body 3D inference that have been out of reach so far. One of them is, how well our current discriminative 2D models learn about the underlying 3D world and what their limits are w.r.t. the number of details of the modeled 3D object. To analyze this, we set up a series of experiments investigating how the representations of segments (few large areas) and keypoints (few small 'dots') converge (many small, distinct 'dots' forming areas). This answers implicitly how well the underlying mesh structure can be learned by a discriminative model.

We created a series of increasingly fine segmented meshes (see Figure 11.2) and top out at 384 segments (in theory, fine-grained mesh segment prediction could be pushed up to vertex level; in practice, however, all current CNN models use a bottleneck representation which limits the size of segments by its resolution). Curiously, up to the maximum number of segments, the prediction performance

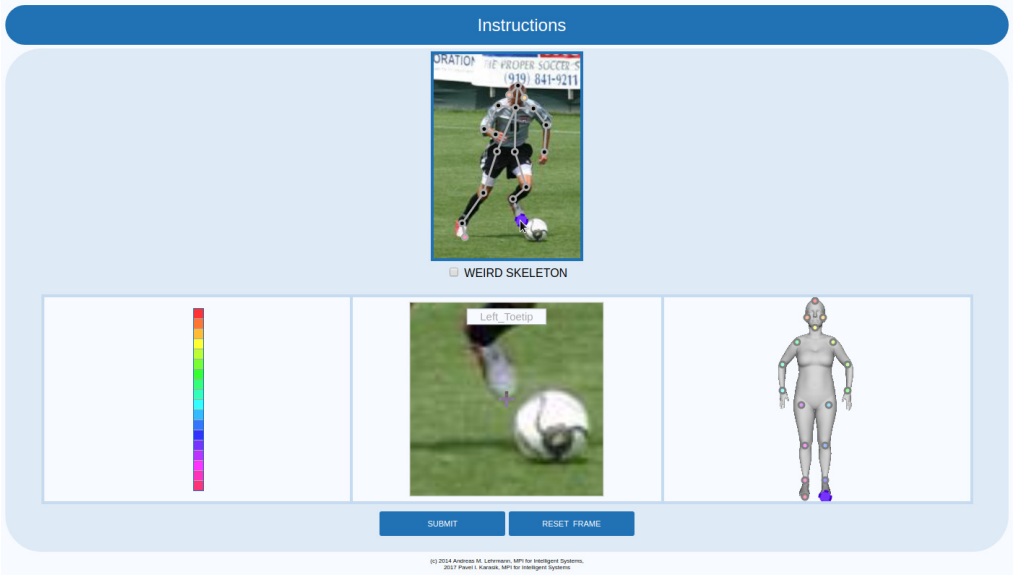
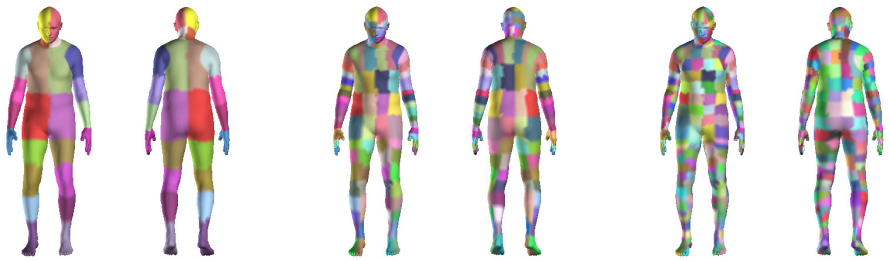


Figure 11.1: *Interface of the proposed keypoint annotation tool.* We leave the original keypoints fixed and let users annotate additional 'hint' keypoints for rotation information. Black points can not be moved by the users and indicate the already marked parts of the skeleton by previous annotators. The users can report problems with this original annotation with an additional checkbox. When hovering with the cursor over a point marker, the point to annotate is highlighted on the 3D body on the lower right side. When dragging a point, a zoom with position indicator is displayed in the lower middle to facilitate accurate annotation. We thank Andreas Lehrmann and Pavel Karasik for their work on the annotation tool.



(a) **(b)** **(c)**

Figure 11.2: *Meshes for the experiments on high numbers of segments.* Ltr.: 24, 96 and 384 segments. Starting with 96 segments, segments are asymmetrical between front and back.

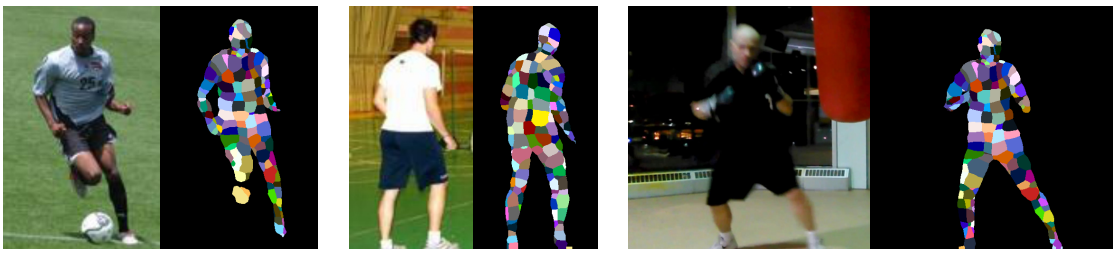


Figure 11.3: *Segmentation results from a 384 segment model.* The model learns the structural connections between segments and produces plausible results. The performance for this model saturates at roughly this size of segments, because the resolution of the inner layers is not sufficient to represent smaller segments.

remains at a high level. Qualitative example results from a 384 segment model can be found in Figure 11.3. We even observe an improving effect when the high segment model predictions are combined to coarser ones: the performance of a model trained to predict 384 segments projected to 12 is better than the performance of a model trained to predict 12 segments from the beginning.

These preliminary results indicate that indeed the 2D discriminative models learn about the 3D nature of their prediction target. The more information about it is carried in the loss function, the better. Additionally, the fine-grained mesh segment predictions are a good starting point for highly detailed 3D pose and shape predictions.

A generative model of people in clothing with 3D bodies. A realistic, probabilistic model of clothed people in 3D would open up a plethora of applications in science and industry. However, the proposed generative model produces solely 2D results. Combining it with the model of clothed people from [PMPHB17], it would be possible to develop a model that predicts 3D pose, shape and clothing from 2D segmentation and pose.

11.1 Outlook

The field around 2D and 3D human modeling is currently in rapid change and moving forward fast, fueled by the improvements of machine learning models.

The traditional 2D human appearance models are evolving fast. In regular intervals, the 2D models are updated with new insights from the machine learning community and new structures specific for human analysis are added and developed, *e.g.*, *Mask R-CNN* [HGDG17] couples traditional 14 keypoint human pose estimation and instance segmentation and two recent approaches [YLO+17, HGT17] achieve notable improvements on the standard benchmarks by incorporating scale into the detection process for keypoints. The models are getting increasingly finetuned for their task. This trend can be expected to still continue a bit, but with the already

very high performance even on challenging scenes, it will saturate eventually. It will be an interesting mark when the first models achieve super-human performance on a challenging human pose estimation task, which can be expected to be taken rather sooner than later. The datasets used for this purpose moved forward from single people, single modality (*e.g.*, either keypoints or segmentation [JE10, EGW+10]) to multiple people and multiple modalities [LMB+14a]. The datasets already contain crowded scenes of high complexity and the models achieve high performance. It will soon be necessary to integrate significantly more details about hands, faces, identity or clothing to challenge the next generation of 2D models. This will be a tedious but rewarding process for which we hope that the methods presented in this thesis may provide useful ideas.

3D human body models are enriched with an increasing amount of details. The SMPL model used throughout this thesis reached a level of fidelity for the human body that is sufficient for most applications even when using only few shape coefficients (*e.g.*, ∼6mm mean absolute vertex error for 10 shape coefficients [LMR+15]). At the same time it is easy to use, even automatically through an API, and can be animated in real-time. With models for hands [RTB17] and faces [LBB+17] on the way of being integrated, the model meets the requirements to accurately capture the full body. The next big step to take is the 3D modeling of clothes in terms of geometry and texture. ClothCap [PMPHB17] is a first step in that direction, however still limited to non-layered and tight clothing and not including a model of texture. With clothing accessories, such a model could fully capture human appearance in every day life. Orthogonal to this, it would be possible to model the anatomical details inside of the body, which could have interesting health related applications.

The methods between 2D and 3D human representation are necessary to tie the 'two worlds' together. This area received far less attention than 'pure' 2D or 3D human modeling. Nevertheless, these combining methods have a tremendous potential, since they allow integrating 2D data (easy to acquire, easy to annotate, available in large quantities) with accurate models and prior beliefs (*e.g.*, beliefs on the human body, on clothing, on identity). We believe that these methods 'between worlds' will eventually drive the development of 'pure' 2D or 3D models with better datasets, refined models and fitting techniques and hope that with the methods and datasets presented in this thesis, we can inspire future research in this direction.

With the potential advances described above, many appealing applications can become reality in the future. An accurate digital avatar of yourself could become a companion through life. It could be updated to match your look and body every morning in front of a virtual 'mirror' and be enriched with medical data to be an accurate digital version of yourself. When shopping online, your browser could ask you—similarly to today, whether it may access your microphone—whether it may access your avatar shape to have a website provide you with perfectly matching clothes and shoes. And doctors could, with privileged access to your encrypted health data, look even at your inner body, accurately enriched by the last MRT/CT

scans through a unified interface: your avatar. Back at home, you can remotely participate in a meeting in virtual reality with your avatar dressed in your clothes and acting and moving just like you do.

This is just one area of applications that could develop out of the presented and related technologies. While there are still many steps to take to make this vision become a reality—in particular, standards need to be developed and implemented to thoroughly protect the privacy of users—they have the potential to simplify and improve our every day life.

Appendices

Appendix A

Equivalence of N2 and the Gini measure

This is a longer version of the proof given in Equation 3.6.

$$
\begin{aligned}
N_2(\boldsymbol{p}) &= \|\boldsymbol{u}-\boldsymbol{e}\|_2^2 - \|\boldsymbol{p}-\boldsymbol{e}\|_2^2 \\
&= \left(1-\frac{1}{W}\right)^2 + (W-1)\left(\frac{1}{W}\right)^2 - \sum_{i=1}^{W}\left|p_i-\frac{1}{W}\right|^2 \\
&= 1-\frac{2}{W}+\frac{1}{W^2}+(W-1)\frac{1}{W^2} - \sum_{i=1}^{W}\left(p_i^2-\frac{2}{W}p_i+\frac{1}{W^2}\right) \\
&= 1-\frac{2}{W}+\frac{1}{W}-\sum_{i=1}^{W}p_i^2+\frac{2}{W}\sum_{i=1}^{W}p_i-\frac{1}{W} \\
&= 1-\frac{1}{W}-\frac{1}{W}+\frac{2}{W}\sum_{i=1}^{W}p_i-\sum_{i=1}^{W}p_i^2 \\
&= 1-\sum_{i=1}^{W}p_i^2. \qquad \text{(A.1)}
\end{aligned}
$$

Appendix B

Metric property of the induced entropy for 0 < q < 1

For $0 < q < 1$, the inducing term loses its property as a norm because the triangle inequality does not hold any more. However, the exponent q restores the triangle inequality, hence N_q still 'works' for these values. To see this, we need to show that $||\boldsymbol{x} + \boldsymbol{y}||_q^q \leq ||\boldsymbol{x}||_q^q + ||\boldsymbol{y}||_q^q$ with $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and $\forall i : x_i, y_i \geq 0$. First, we note that

$$||x + y||_q^q = \sum_{i=1}^{n} |x_i + y_i|^q = \sum_{i=1}^{n} (x_i + y_i)^q. \tag{B.1}$$

In a next step, we need to show that $(x_i + y_i)^q \leq (x_i)^q + (y_i)^q$. The case of equality holds exactly if one of x_i or y_i are 0; trivially, the equation also holds if both are 0. For all other cases, similar to the proof in [Ros12], we can define a function $f(t) := (1 + t)^q - 1 - t^q$ for $t \geq 0$. Then $f'(t) = q(1 + t)^{q-1} - qt^{q-1} < 0$ for all $t \in [0, \infty[$. Since $f(0) = 0$, it follows that $f(t) < 0$ on $[0, \infty[$. Substituting $t = \frac{x_i}{y_i}$,

$$\begin{aligned} \left(1 + \frac{x_i}{y_i}\right)^q - 1 - \left(\frac{x_i}{y_i}\right)^q < 0 \quad &\Leftrightarrow \quad \left(\frac{x_i + y_i}{y_i}\right)^q - 1 - \left(\frac{x_i}{y_i}\right)^q < 0 \\ &\Leftrightarrow \quad (x_i + y_i)^q - (x_i^q + y_i^q) < 0 \\ &\Leftrightarrow \quad (x_i + y_i)^q < x_i^q + y_i^q. \end{aligned} \tag{B.2}$$

Since f is strictly decreasing, equality holds only if one of the two variables is zero. With this, it follows that

$$\begin{aligned}\sum_{i=1}^{n}(x_i+y_i)^q &\leq \sum_{i=1}^{n}(x_i^q+y_i^q)\\ &= \sum_{i=1}^{n}x_i^q+\sum_{i=1}^{n}y_i^q\\ &= ||\boldsymbol{x}||_q^q+||\boldsymbol{y}||_q^q. \end{aligned} \tag{B.3}$$

Appendix C

Integral of the Normal Distribution to Power

This is the derivation for Equation 3.14. We show the derivation for the univariate case, the derivation for the multivariate case follows similarly by using the definition of the multivariate normal distribution.

$$\begin{aligned}
\int_{-\infty}^{\infty} \mathcal{N}(x;\mu,\sigma)^q dx &= \int_{-\infty}^{\infty} (\sqrt{2\pi}\sigma)^{-q} e^{-q\frac{(x-\mu)^2}{\sigma^2}} dx \\
&= \int_{-\infty}^{\infty} (\sqrt{2\pi}\sigma)^{-q} \cdot \frac{\sqrt{2\pi}\sigma/\sqrt{q}}{\sqrt{2\pi}\sigma/\sqrt{q}} e^{\frac{(x-\mu)^2}{(\sigma/\sqrt{q})^2}} dx \\
&= \frac{1}{\sqrt{q}}\sqrt{2\pi}\sigma \cdot (\sqrt{2\pi}\sigma)^{-q} \cdot \int_{-\infty}^{\infty} \mathcal{N}(x;\mu,\frac{\sigma}{\sqrt{q}}) dx \\
&= \frac{1}{\sqrt{q}}(\sqrt{2\pi}\sigma)^{-(q-1)}. \quad \square
\end{aligned} \tag{C.1}$$

Appendix D

Segmentation Fusion Model Hyperparameters

To determine the best network architecture and configuration for the fusion model, we did a parameter sweep over the most important hyperparameters. The results can be found in Figure D.1. For the search, we trained all networks for 20k iterations and take the maximum accuracy as an indicator on how well they will perform after convergence. We noticed that at this point of the training, the final performance could be estimated well. The other hyperparameters were kept fixed, most notably, the learning rate. We used the adaptive ADAM solver [KB14] to account for smaller deviations in the preferred network learning rate.

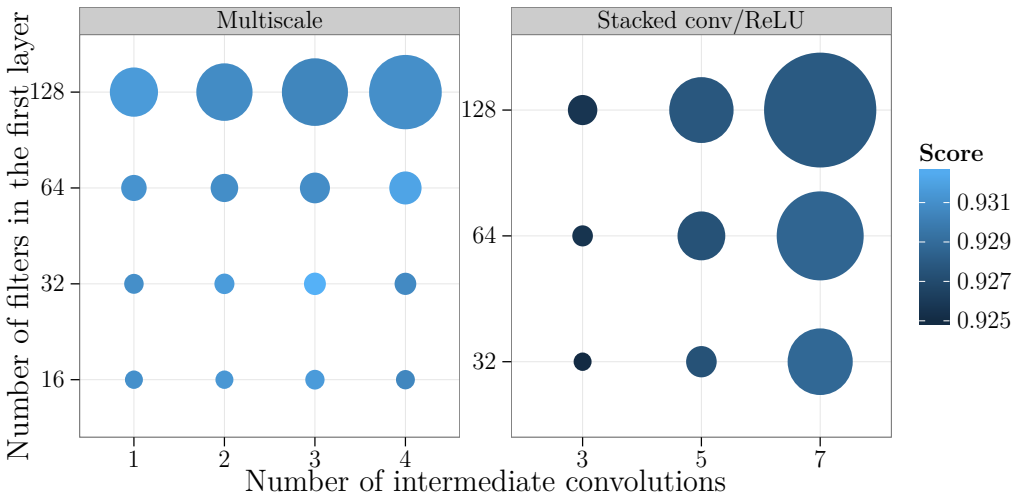


Figure D.1: *Hyperparameter search for the fusion models (validation set scores).* The size of the circles is proportional to the model runtime. The highest runtime is 12.5 times higher than the lowest. The best scoring architecture is also amongst the fastest.

Appendix E

Additional Example Results for UP Models

We provide additional examples from a diverse set of datasets for all our discussed learning and fitting methods in Figure E.1. We added examples from the Fashionista dataset [YKOB12], from which we did not use data for finetuning the models. Further samples from the LSP dataset [JE10] and the MPII-HPDB [APGS14] are provided, which have harder to parse backgrounds than the motion capture datasets.

(a) 31 Part Segmentation.

(b) 91 Keypoint Pose Estimation.

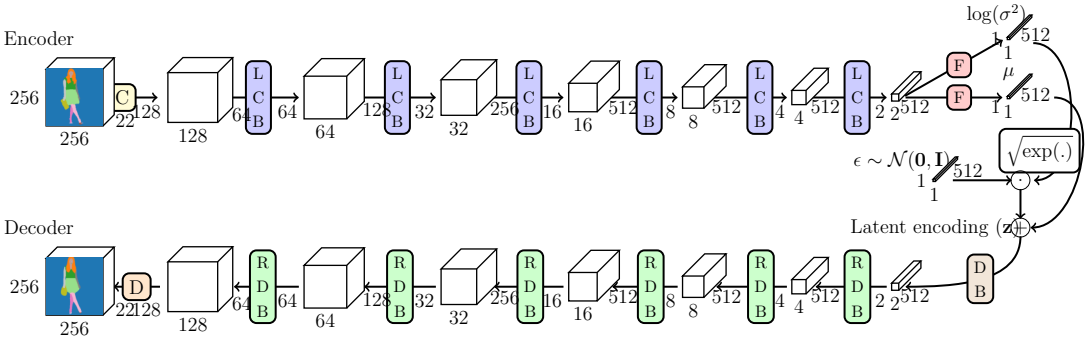
(c) 3D Fitting on 91 Landmarks.

(d) Direct 3D Pose and Shape Prediction.

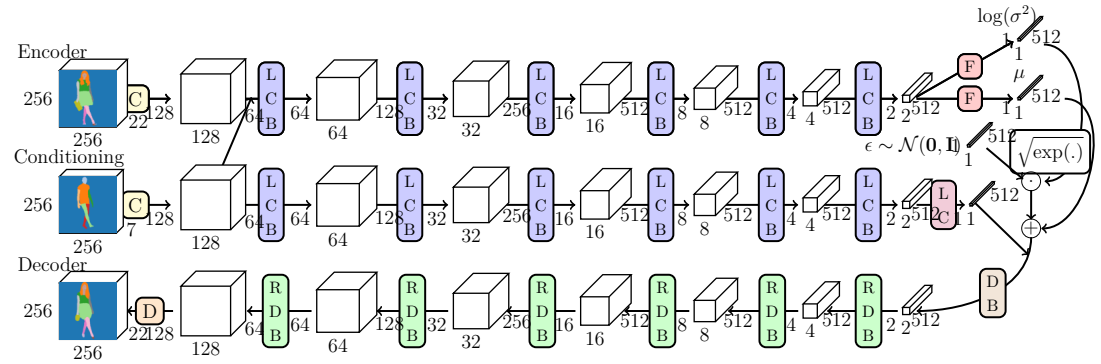
Figure E.1: *Additional results on the Fashionista dataset [YKOB12] (first three images),* LSP *dataset [JE10] (images three to six) and* MPII-HPDB *[APGS14] (last two images).*

Appendix F

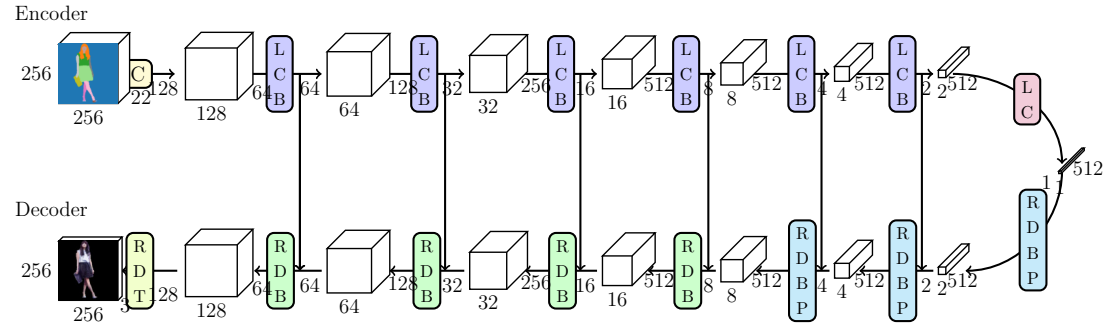
ClothNet CNN Architectures



(a) The *latent sketch module.*



(b) The *conditional sketch module.*



(c) The *portrait module.*

Figure F.1: *Full architectures of the generative* CNN *models.* Data size proportions scale logarithmically. Joining arrows indicate concatenation along the filter axis. The transformation blocks are: C 4×4 convolution, LCB LReLU, 4×4 convolution, batchnorm, F fully connected layer, DB 4×4 deconvolution, batchnorm; RDB ReLU, 4×4 deconvolution, batchnorm; D 4×4 deconvolution; LC LReLU, 4×4 convolution; RDBP ReLU, deconvolution, batchnorm, dropout; RDT ReLU, deconvolution, tanh.

Lists and References

Acronyms

AMT Amazon Mechanical Turk.

ANN Artificial Neural Network.

AUC Area Under Curve.

CDF Cumulative Distribution Function.

CNN Convolutional Neural Network.

CRF Conditional Random Field.

CVAE Conditional Variational AutoEncoder.

FPR False Positive Rate.

GMM Gaussian Mixture Model.

HMI Human Machine Interaction.

HOG Histogram of oriented Gradients.

HPC High Performance Computing.

IOU Intersection over Union.

KL Kullback-Leibler.

LRELU Leaky ReLU.

LSP Leeds Sports Pose.

MAP Most probable A Posteriori.

MPII-HPDB MPII Human Pose Database.

MSE Mean Squared Error.

PCA Principal Component Analysis.

RELU Rectified Linear Unit.

ROC Receiver Operator Characteristic.

SGD Stochastic Gradient Descent.

SMPL Skinned Multi Person Linear model.

TPR True Positive Rate.

VAE Variational AutoEncoder.

List of Tables

List of Figures

Bibliography

A

[AAB+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia Yangqing, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. `http://tensorflow.org/`, 2015. (Cited on page 46.)

[AAD07] Pedram Azad, Tamim Asfour, and Rudiger Dillmann. Toward an Unified Representation for Imitation of Human Motion on Humanoids. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 2558–2563, apr 2007. (Cited on page 79.)

[AB15] Ijaz Akhter and Michael J. Black. Pose-conditioned joint angle limits for 3D human pose reconstruction. *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1446–1455, 2015. (Cited on pages 7 and 59.)

[Abe02] Sumiyoshi Abe. Stability of Tsallis entropy and instabilities of Rényi and normalized Tsallis entropies: A basis for q-exponential distributions. *Phys. Rev. E*, 66(4):46134, 2002. (Cited on page 27.)

[APGS14] Mykhaylo Andriluka, Leonid Pishchulin, Peter Vincent Gehler, and Bernt Schiele. 2D Human Pose Estimation: New Benchmark and State of the Art Analysis. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014. (Cited on pages 7, 65, 66, 68, 77, 78, 82, 99, 135, and 136.)

[ARS09] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. Pictorial Structures Revisited: People Detection and Articulated Pose Estimation. *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009. (Cited on pages 7 and 63.)

[ASK+05] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. SCAPE: Shape Comple-

tion and Animation of PEople. *ACM Trans. Graph. - Proc. ACM SIGGRAPH*, 24(3):408–416, 2005. (Cited on pages 19, 62, and 97.)

B

[BF11] Yihang Bo and Charless C. Fowlkes. Shape-based pedestrian parsing. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 2265–2272, 2011. (Cited on page 63.)

[BFSO84] Leo Breiman, Jerome Friedman, Charles J. Stone, and Richard A. Olshen. *Classification and regression trees*. CRC press, 1984. (Cited on page 13.)

[BJ13] Yihang Bo and Hao Jiang. Scale and Rotation Invariant Approach to Tracking Human Body Part Regions in Videos. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1041–1047, 2013. (Cited on page 63.)

[BKL+16a] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Vincent Gehler, Javier Romero, and Michael J. Black. Keep it SMPL: Automatic Estimation of 3D Human Pose and Shape from a Single Image. `https://link.springer.com/chapter/10.1007/978-3-319-46454-1_34`, 2016. (Cited on pages 11, 55, 56, 57, 58, 59, 62, 78, 79, 87, and 88.)

[BKL+16b] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Vincent Gehler, Javier Romero, and Michael J. Black. SMPLify website. `http://smplify.is.tue.mpg.de`, 2016. (Cited on page 10.)

[BL13] K. Bache and M. Lichman. UCI Machine Learning Repository. `http://archive.ics.uci.edu/ml`, 2013. (Cited on pages 32, 33, 43, and 44.)

[BM09] Lubomir Bourdev and Jitendra Malik. Poselets: Body Part Detectors Trained Using 3D Human Pose Annotations. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1365–1372. IEEE, 2009. (Cited on pages 64 and 78.)

[Boc] Sergey Bochkanov. ALGLIB. `http://www.alglib.net`. (Cited on page 40.)

[Bra] Gary Rost Bradski. OpenCV. `http://www.opencv.org`. (Cited on page 40.)

[Bre01] Leo Breiman. Random Forests. *Mach. Learn.*, 45(1):5–32, 2001. (Cited on page 13.)

[Bre04] Alexandra P. Bremner. *Localised splitting criteria for classification and regression trees*. PhD thesis, Murdoch University, 2004. (Cited on page 24.)

[BU02] Eran Borenstein and Shimon Ullman. Class-specific, top-down segmentation. In *Proc. Eur. Conf. Comput. Vis.*, pages 109–122, 2002. (Cited on pages 34 and 35.)

[BUSB13] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. OpenSurfaces: A Richly Annotated Catalog of Surface Appearance. *ACM Trans. Graph. - Proc. ACM SIGGRAPH*, 32(4), 2013. (Cited on page 65.)

C

[CF08] Cheng Chen and Guoliang Fan. Hybrid Body Representation for Integrated Pose Recognition, Localization and Segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2008. (Cited on page 63.)

[CKC10] Yu Chen, Tae-Kyun Kim, and Roberto Cipolla. Inferring 3D shapes and deformations from single views. In *Proc. Eur. Conf. Comput. Vis.*, pages 300–313, 2010. (Cited on pages 7 and 62.)

[CLL+15] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. `http://mxnet.rtfd.org`, 2015. (Cited on page 46.)

[CLZ13] Yinpeng Chen, Zisheng Liu, and Zhengyou Zhang. Tensor-based human body modeling. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 105–112, 2013. (Cited on page 19.)

[CML+14] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. Detect What You Can: Detecting and Representing Objects using Holistic Models and Body Parts. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014. (Cited on pages 7, 64, and 79.)

[CMU] CMU Graphics Lab. CMU Mocap database. `http://mocap.cs.cmu.edu`. (Cited on page 57.)

[CPK+15] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. In *Proc. Int. Conf. Learn. Represent.*, 2015. (Cited on pages 7, 108, and 115.)

[CPM+16] James Charles, Tomas Pfister, Derek Magee, David Hogg, and Andrew Zisserman. Personalizing Human Video Pose Estimation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016. (Cited on page 78.)

[CS13] Antonio Criminisi and Jamie Shotton, editors. *Decision Forests for Computer Vision and Medical Image Analysis*. Springer-Verlag London, 2013. (Cited on pages 13, 24, 30, 39, and 40.)

[CSK11] Antonio Criminisi, Jamie Shotton, and Ender Konukoglu. Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning. Technical Report MSR-TR-2011-114, MS Res., 2011. (Cited on page 40.)

[CSZ06] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors. *Semi-Supervised Learning*. MIT Press, 2006. (Cited on pages 32 and 44.)

[CWL+16] Wenzheng Chen, Huan Wang, Yangyan Li, Hao Su, Zhenhua Wang, Changhe Tu, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. Synthesizing Training Images for Boosting Human 3D Pose Estimation. In *Proc. Int. Conf. 3D Vis.*, 2016. (Cited on pages 96 and 97.)

[Cyb89] George Cybenko. Approximation by superpositions of a sigmoidal function. *Math. Control. signals Syst.*, 2(4):303–314, 1989. (Cited on page 15.)

[CYW+16] Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu, and Alan L. Yuille. Attention to Scale: Scale-aware Semantic Image Segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016. (Cited on page 83.)

D

[dASTH10] Edilson de Aguiar, Leonid Sigal, Adrien Treuille, and Jessica K. Hodgins. Stable Spaces for Real-time Clothing. *ACM Trans. Graph. - Proc. ACM SIGGRAPH*, 29(4):106:1–106:9, 2010. (Cited on pages 7 and 97.)

[dCBV09] Teofilio E. de Campos, Bodla Rakesh Babu, and Manik Varma. Character recognition in natural images. In *Proc. Int. Conf. Comput. Vis. Theory Appl.*, 2009. (Cited on pages 32, 43, and 44.)

[DCH+16] Jian Dong, Qiang Chen, Zhongyang Huang, Jinchao Yang, and Shuicheng Yan. Parsing Based on Parselets: A Unified Deformable Mixture Model for Human Parsing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(1), 2016. (Cited on page 63.)

[DCSF15] Emily Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks. In *Adv. Neural Inf. Process. Syst.*, pages 1486–1494, 2015. (Cited on page 98.)

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-jia Li, Kai Li, and Li Fei-fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009. (Cited on page 15.)

[DGL14] Matthias Dantone, Jürgen Gall, and Christian Leistner. Body Parts Dependent Joint Regressors for Human Pose Estimation in Still Images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2014. (Cited on pages 7, 35, 44, 82, and 86.)

[DGLG13] Matthias Dantone, Jürgen Gall, Christian Leistner, and Luc Van Gool. Human Pose Estimation using Body Parts Dependent Joint Regressors. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013. (Cited on pages 34, 35, and 36.)

[DQN16] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape Completion using 3D-Encoder-Predictor CNNs and Shape Synthesis. *ArXiv e-prints*, 2016, 1612.00101. (Cited on page 98.)

[DSR+15] Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, Daniel Nouri, Daniel Maturana, Martin Thoma, Eric Battenberg, Jack Kelly, Jeffrey De Fauw, Michael Heilman, Diogo149, Brian McFee, Hendrik Weideman, Takacsg84, Peterderivaz, Jon, Instagibbs, Kashif Rasul, CongLiu, Britefury, and Jonas Degrave. Lasagne: First release. `http://dx.doi.org/10.5281/zenodo.27878`, 2015. (Cited on page 46.)

[DT05] Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for Human Detection. *IEEE Conf. Comput. Vis. Pattern Recognit.*, 1:886–893, 2005. (Cited on page 63.)

E

[EdAJ+15] Ahmed Elhayek, Edilson de Aguiar, Arjun Jain, Jonathan Tompson, Leonid Pishchulin, Micha Andriluka, Chris Bregler, Bernt Schiele, and Christian Theobalt. Efficient ConvNet-based marker-less motion capture in general scenes with a low number of cameras. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 3810–3818. IEEE, 2015. (Cited on page 79.)

[EGW+10] Mark Everingham, Luc Van Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.*, 88(2):303–338, 2010. (Cited on pages 7, 64, 79, and 122.)

F

[FH00] Pedro Felipe Felzenszwalb and Daniel P. Huttenlocher. Efficient Matching of Pictorial Structures. *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2:66–73, 2000. (Cited on page 63.)

[FH05] Pedro F Felzenszwalb and Daniel P Huttenlocher. Pictorial Structures for Object Recognition. *Int. J. Comput. Vis.*, 61(1):55–79, 2005. (Cited on pages 7 and 63.)

[FMJZ08] Vittorio Ferrari, Manuel Marín-Jiménez, and Andrew Zisserman. Progressive Search Space Reduction for Human Pose Estimation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2008. (Cited on pages 63 and 64.)

[Fri01] Jerome H. Friedman. Greedy function approximation: a gradient boosting machine. *Ann. Stat.*, pages 1189–1232, 2001. (Cited on page 14.)

[Fuk80] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybern.*, 36:193–202, 1980. (Cited on page 16.)

[FWZB10] Oren Freifeld, Alexander Weiss, Silvia Zuffi, and Michael J. Black. Contour people: A parameterized model of 2D articulated human shape. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 639–646. IEEE, 2010. (Cited on page 79.)

G

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. (Cited on page 17.)

[GDG+15] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. DRAW: A recurrent neural network for image generation. *ArXiv e-prints*, 2015, 1502.04623. (Cited on page 98.)

[GFB10] Peng Guan, Oren Freifeld, and Michael J. Black. A 2D human body model dressed in eigen clothing. In *Proc. Eur. Conf. Comput. Vis.*, pages 285–298. Springer, 2010. (Cited on page 97.)

[GHF+07] Rony Goldenthal, David Harmon, Raanan Fattal, Michel Bercovier, and Eitan Grinspun. Efficient Simulation of Inextensible Cloth. *ACM Trans. Graph. - Proc. ACM SIGGRAPH*, 26(3), 2007. (Cited on page 97.)

[GM87] Stuart Geman and Donald E. McClure. Statistical methods for tomographic image reconstruction. *Bull. Int. Stat. Inst.*, 52(4), 1987. (Cited on page 56.)

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Adv. neural Inf. Process. Syst.*, pages 2672–2680, 2014. (Cited on page 98.)

[GRH+12] Peng Guan, Loretta Reiss, David A. Hirshberg, Alexander Weiss, and Michael J. Black. DRAPE: DRessing Any PErson. *ACM Trans. Graph. - Proc. ACM SIGGRAPH*, 31(4):35:1—-35:10, 2012. (Cited on pages 7 and 97.)

[Gua12] Peng Guan. *Virtual Human Bodies with Clothing and Hair: From Images to Animation*. PhD thesis, Brown University, Providence, Rhode Island, USA, 2012. (Cited on pages 7 and 62.)

[GWBB09] Peng Guan, Alexander Weiss, Alexandru O. Balan, and Michael J. Black. Estimating human shape and pose from a single image. *Proc. IEEE Int. Conf. Comput. Vis.*, pages 1381–1388, 2009. (Cited on pages 7, 62, and 79.)

[GYR+11] Jürgen Gall, Angela Yao, Nima Razavi, Luc Van Gool, and Victor Lempitsky. Hough Forests for Object Detection, Tracking and Action Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(11):2188–2202, 2011. (Cited on page 34.)

H

[HAR+10] Nils Hasler, Hanno Ackermann, Bodo Rosenhahn, Thorsten Thormählen, and Hans-Peter Seidel. Multilinear pose and body shape estimation of dressed subjects from image sets. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1823–1830, 2010. (Cited on pages 7, 19, 62, and 79.)

[HBL+17] Yinghao Huang, Federica Bogo, Christoph Lassner, Angjoo Kanazawa, Peter Vincent Gehler, Javier Romero, Ijaz Akhter, and Michael J. Black. Towards Accurate Marker-less Human Shape and Pose Estimation over Time. In *Proc. Int. Conf. 3D Vis.*, 2017. (Cited on page 12.)

[HFH+09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The WEKA Data Mining Software: An Update. *SIGKDD Expl.*, 11(1), 2009. (Cited on page 40.)

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proc. Int. Conf. Comput. Vis.*, pages 2961–2969, 2017. (Cited on page 121.)

[HGT17] Shaoli Huang, Mingming Gong, and Dacheng Tao. A Coarse-Fine Network for Keypoint Localization. In *Proc. Int. Conf. Comput. Vis.*, pages 3028–3037, 2017. (Cited on page 121.)

[Hog83] David Hogg. Model-based vision: a program to see a walking person. *Image Vis. Comput.*, 1(1):5–20, 1983. (Cited on page 79.)

[Hor91] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks1*, 4(2):251–257, 1991. (Cited on page 15.)

[Hul94] Jonathan J. Hull. A Database for Handwritten Text Recognition Research. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(5):550–554, 1994. (Cited on pages 32, 43, and 44.)

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proc. Int. Conf. Comput. Vis.*, pages 1026–1034, 2015. (Cited on pages 45 and 70.)

[HZRS16a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 770–778, 2016. (Cited on pages 17 and 45.)

[HZRS16b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity Mappings in Deep Residual Networks. *ArXiv e-prints*, 2016, 1603.05027. (Cited on pages 45 and 115.)

I

[ICS14] Catalin Ionescu, Joao Carreira, and Cristian Sminchisescu. Iterated second-order label sensitive pooling for 3D human pose estimation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1661–1668, 2014. (Cited on page 79.)

[IMA+16] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. *ArXiv e-prints*, 2016, 1602.07360. (Cited on pages 18 and 90.)

[IPA+16] Eldar Insafutdinov, Leonid Pishchulin, Björn Andres, Mykhaylo Andriluka, and Bernt Schiele. DeeperCut: A Deeper, Stronger, and Faster Multi-Person Pose Estimation Model. In *Proc. Eur. Conf. Comput. Vis.*, 2016. (Cited on pages 6, 7, 78, 85, 86, and 99.)

[IPOS14] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2014. (Cited on pages 55, 58, 64, 67, 68, 69, 78, 96, 97, and 99.)

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ArXiv e-prints*, 2015, 1502.03167. (Cited on pages 17, 67, 70, and 104.)

[IZZE16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *ArXiv e-prints*, 2016, 1611.07004. (Cited on pages 98, 100, 103, and 109.)

J

[JBY96] Shanon X. Ju, Michael J. Black, and Yaser Yacoob. Cardboard people: A parameterized model of articulated motion. In *Proc. Int. Conf. Autom. Face- Gesture-Recognition*, pages 38–44, Killington, Vermont, 1996. (Cited on page 79.)

[JE10] Sam Johnson and Mark Everingham. Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation. In *Proc. Br. Mach. Vis. Conf.*, pages 12.1–12.11. BMVA Press, 2010. (Cited on pages 7, 34, 55, 56, 58, 65, 66, 68, 78, 82, 99, 122, 135, and 136.)

[JE11] Sam Johnson and Mark Everingham. Learning effective human pose estimation from inaccurate annotation. *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 1465–1472, 2011. (Cited on pages 65, 66, 78, and 82.)

[JGZ+13] Hueihan Jhuang, Jürgen Gall, Silvia Zuffi, Cordelia Schmid, and Michael J. Black. Towards understanding action recognition. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 3192–3199, Sydney, Australia, 2013. IEEE. (Cited on page 79.)

[JSD+14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. In *Proc. ACM Multimed.*, 2014. (Cited on pages 10, 39, and 45.)

[JTST10] Arjun Jain, Thorsten Thormählen, Hans-Peter Seidel, and Christian Theobalt. MovieReshape: Tracking and Reshaping of Humans in Videos. *ACM Trans. Graph. - Proc. ACM SIGGRAPH Asia*, 29(5), 2010. (Cited on page 97.)

K

[KB14] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *ArXiv e-prints*, 2014, 1412.6980. (Cited on pages 15, 48, 104, and 133.)

[KFCR15] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulò. Deep Neural Decision Forests. In *Proc. Int. Conf. Comput. Vis.*, 2015. (Cited on page 14.)

[KGBS11] Ladislav Kavan, Dan Gerszewski, Adam W. Bargteil, and Peter-Pike Sloan. Physics-inspired Upsampling for Cloth Simulation in Games. *ACM Trans. Graph. - Proc. ACM SIGGRAPH*, 30(4):93:1–93:10, 2011. (Cited on page 97.)

[Khi57] Aleksandr Yakovlevich Khinchin. *Mathematical Foundations of Information Theory*. Courier Corporation, 1957. (Cited on page 24.)

[Kin09] Davis E. King. Dlib-ml: A Machine Learning Toolkit. *J. Mach. Learn. Res.*, 10:1755–1758, 2009. (Cited on page 100.)

[KKN+13] Doyub Kim, Woojong Koh, Rahul Narain, Kayvon Fatahalian, Adrien Treuille, and James F. O'Brien. Near-exhaustive Precomputation of Secondary Cloth Effects. *ACM Trans. Graph. - Proc. ACM SIGGRAPH*, 32(4):87:1–7, 2013. (Cited on pages 7 and 97.)

[KMRW14] Diederik P. Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Adv. Neural Inf. Process. Syst.*, pages 3581–3589, 2014. (Cited on pages 98 and 103.)

[KRBT08] Pushmeet Kohli, Jonathan Rihan, Matthieu Bray, and Philip H. S. Torr. Simultaneous Segmentation and Pose Estimation of Humans using Dynamic Graph Cuts. *Int. J. Comput. Vis.*, 79:285–298, 2008. (Cited on page 63.)

[KS14] Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1867–1874, 2014. (Cited on page 100.)

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Everest Hinton. Imagenet classification with deep convolutional neural networks. In *Adv. Neural Inf. Process. Syst.*, volume 25, pages 1097–1105, 2012. (Cited on pages 15 and 17.)

[KW13] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *ArXiv e-prints*, 2013, 1312.6114. (Cited on pages 98 and 102.)

[KWKT15] Tejas D. Kulkarni, William F. Whitney, Pushmeet Kohli, and Joshua B. Tenenbaum. Deep convolutional inverse graphics network. In *Adv. Neural Inf. Process. Syst.*, pages 2539–2547, 2015. (Cited on page 98.)

L

[LB14] Matthew M. Loper and Michael J. Black. OpenDR: An Approximate Differentiable Renderer. In *Proc. Eur. Conf. Comput. Vis.*, pages 154–169, Cham, 2014. Springer International Publishing. (Cited on pages 19, 58, and 74.)

[LBB+17] Tianye Li, Timo Bolkart, Michael J. Black, Hao Li, and Javier Romero. Learning a Model of Facial Shape and Expression from 4D Scans. *ACM Trans. Graph. - Proc. ACM SIGGRAPH Asia*, 36(6):194:1–194:17, 2017. (Cited on page 122.)

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998. (Cited on page 16.)

[LC85] Hsi-Jian Lee and Zen Chen. Determination of 3D human body postures from a single view. *Comput. Vis. Graph. Image Process.*, 30(2), 1985. (Cited on page 62.)

[LC14] Sijin Li and Antoni B. Chan. 3D Human Pose Estimation from Monocular Images with Deep Convolutional Neural Network. In *Proc. Asian Conf. Comput. Vis.*, pages 332–347, 2014. (Cited on page 79.)

[Les82] Bernhard Lesche. Instabilities of Rényi entropies. *J. Stat. Phys.*, 27(2):419–422, 1982. (Cited on page 27.)

[LFSL12] Huchuan Lu, Goliang Fang, Xinqing Shao, and Xuelong Li. Segmenting Human From Photo Images Based on a Coarse-to-Fine Scheme. *IEEE Trans. Syst. Man Cybern.*, 42(3):889–899, 2012. (Cited on page 63.)

[LHHH13] Taegyu Lim, Seunghoon Hong, Bohyung Han, and Joon Hee Han. Joint Segmentation and Pose Tracking of Human in Natural Videos. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 833–840, 2013. (Cited on page 63.)

[LKKG15] Christoph Lassner, Daniel Kappler, Martin Kiefel, and Peter Vincent Gehler. barrista Homepage. `http://classner.github.io/barrista/`, 2015. (Cited on page 10.)

[LKKG16] Christoph Lassner, Daniel Kappler, Martin Kiefel, and Peter Vincent Gehler. Barrista - Caffe Well-Served. `http://doi.acm.org/10.1145/2964284.2973803`, 2016. (Cited on pages 11, 47, and 48.)

[LL13] Christoph Lassner and Rainer Lienhart. Methods and applications for distance based ANN training. In *Proc. Int. Conf. Mach. Learn. Appl.*, volume 2, 2013. (Cited on page 11.)

[LL14] Christoph Lassner and Rainer Lienhart. fertilized forests Homepage. `http://www.fertilized-forests.org`, 2014. (Cited on page 10.)

[LL15a] Christoph Lassner and Rainer Lienhart. Norm-induced entropies for decision forests. In *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2015. (Cited on pages 11, 27, 29, 32, 33, 34, 35, and 36.)

[LL15b] Christoph Lassner and Rainer Lienhart. The fertilized forests decision forest library. `http://doi.acm.org/10.1145/2733373.2807407`, 2015. (Cited on pages 11, 40, 42, and 43.)

[LMB+14a] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *Proc. Eur. Conf. Comput. Vis.*, pages 740–755, Cham, 2014. Springer International Publishing. (Cited on pages 64, 119, and 122.)

[LMB14b] Matthew M. Loper, Naureen Mahmood, and Michael J. Black. MoSh: Motion and shape capture from sparse markers. *ACM Trans. Graph. - Proc. ACM SIGGRAPH Asia*, 33(6), 2014. (Cited on page 79.)

[LMR+15] Matthew M. Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graph. - Proc. ACM SIGGRAPH Asia*, 34(6), 2015. (Cited on pages 18, 19, 55, 57, 80, 99, and 122.)

[LN09] Mun Wai Lee and Ramakant Nevatia. Human Pose Tracking in Monocular Sequence Using Multilevel Structured Models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(1):27–38, 2009. (Cited on page 63.)

[Lop15] Matt Loper. Chumpy. `http://chumpy.org`, 2015. (Cited on pages 19 and 58.)

[LPMG17a] Christoph Lassner, Gerard Pons-Moll, and Peter Vincent Gehler. A Generative Model of People in Clothing. In *Proc. Int. Conf. Comput. Vis.*, pages 853–862, 2017. (Cited on pages 11, 95, 96, 99, 100, 101, 104, 105, 106, 107, 108, and 109.)

[LPMG17b] Christoph Lassner, Gerard Pons-Moll, and Peter Vincent Gehler. A Generative Model of People in Clothing website. `http://files.is.tuebingen.mpg.de/classner/gp`, 2017. (Cited on page 11.)

[LRK+16] Christoph Lassner, Javier Romero, Martin Kiefel, Federica Bogo, Michael J. Black, and Peter Vincent Gehler. Unite the People Website. `http://up.is.tuebingen.mpg.de`, 2016. (Cited on page 10.)

[LRK+17] Christoph Lassner, Javier Romero, Martin Kiefel, Federica Bogo, Michael J. Black, and Peter Vincent Gehler. Unite the People: Closing the Loop between 3D and 2D Human Representations. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017. (Cited on pages 11, 73, 77, 81, 82, 86, 87, 88, and 91.)

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015. (Cited on page 17.)

[LSX13] Huchuan Lu, Xinqing Shao, and Yi Xiao. Pose Estimation With Segmentation Consistency. *IEEE Trans. Image Process.*, 22(10):4040–4048, 2013. (Cited on page 63.)

[LTZ13] L'ubor Ladický, Philip H. S. Torr, and Andrew Zisserman. Human Pose Estimation using a Joint Pixel-wise and Part-wise Formulation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 3378–3585, 2013. (Cited on page 63.)

[LXS+15] Xiaodan Liang, Chunyan Xu, Xiaohui Shen, Jianchao Yang, Si Liu, Jinhui Tang, Liang Lin, and Shuicheng Yan. Human Parsing with

Contextualized Convolutional Neural Network. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 1386–1394, 2015. (Cited on pages 7, 64, 68, 99, and 100.)

M

[Mas05] Marco Masi. A step beyond Tsallis and Rényi entropies. *Phys. Lett. A*, 338(3–5):217–224, 2005. (Cited on page 25.)

[MBLH17] Maren Mahsereci, Lukas Balles, Christoph Lassner, and Philipp Hennig. Early Stopping without a Validation Set. *ArXiv e-prints*, 2017, 1703.09580. (Cited on page 12.)

[MD08] Tomasz Maszczyk and Włodzisław Duch. Comparison of Shannon, Renyi and Tsallis Entropy Used in Decision Trees. In Leszek Rutkowski, Ryszard Tadeusiewicz, Lotfi A. Zadeh, and Jacek M. Zurada, editors, *Artif. Intell. Soft Comput.*, volume 5097 of *Lecture Notes in Computer Science*, pages 643–651. Springer Berlin Heidelberg, 2008. (Cited on page 24.)

[MDFFF17] Seyed-Mohsen Moosave-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. *ArXiv e-prints*, 2017, 1610.08401. (Cited on page 18.)

[MHN13] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. Int. Conf. Mach. Learn.*, volume 30, 2013. (Cited on pages 16 and 104.)

[Mil05] Milopeng. Yoga pose, unchanged. License: `https://creativecommons.org/licenses/by-nc/2.0/`. `https://flic.kr/p/4nv4G`, 2005. (Cited on page 9.)

[MN78] David Marr and H. Keith Nishihara. Representation and Recognition of the Spatial Organization of Three-Dimensional Shapes. *R. Soc. London B Biol. Sci.*, 200(1140):269–294, 1978. (Cited on page 79.)

[MR10] Oded Maimon and Lior Rokach, editors. *Data Mining and Knowledge Discovery Handbook*. Springer, 2nd edition, 2010. (Cited on page 24.)

[MREM04] Greg Mori, Xiaofeng Ren, Alexei A. Efros, and Jitendra Malik. Recovering human body configurations: combining segmentation and recognition. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, volume 2, pages 326–333, 2004. (Cited on page 63.)

[MTD+15] Christian Mandery, Oemer Terlemez, Martin Do, Nikolaus Vahrenkamp, and Tamim Asfour. The KIT Whole-Body Human Motion Database. In *Proc. Int. Conf. Adv. Robot.*, pages 329–336, 2015. (Cited on page 79.)

N

[NB73] Ramakant Nevatia and Thomas O. Binford. Structured Descriptions of Complex Objects. In *Proc. Int. Jt. Conf. Artif. Intell.*, pages 641–647, San Francisco, CA, USA, 1973. Morgan Kaufmann Publishers Inc. (Cited on page 79.)

[NHH15] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning Deconvolution Network for Semantic Segmentation. In *Proc. IEEE Int. Conf. Comput. Vis.*, 2015. (Cited on page 67.)

[NSO12] Rahul Narain, Armin Samii, and James F O'Brien. Adaptive Anisotropic Remeshing for Cloth Simulation. *ACM Trans. Graph. - Proc. ACM SIGGRAPH*, 31(6):147:1–10, nov 2012. (Cited on page 97.)

[NW99] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Verlag, 1999. (Cited on page 58.)

[NYD16] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *Proc. Eur. Conf. Comput. Vis.*, pages 483–499. Springer, 2016. (Cited on pages 7, 78, 85, and 99.)

O

[OVB+16] Gabriel L. Oliveira, Abhinav Valada, Claas Bollen, Wolfram Burgard, and Thomas Brox. Deep learning for human part discovery in images. In Danica Kragic, Antonio Bicchi, and Alessandro De Luca 0001, editors, *Proc. IEEE Int. Conf. Robot. Autom.*, pages 1634–1641. IEEE, 2016. (Cited on pages 7 and 79.)

[OWL15] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Training a feedback loop for hand pose estimation. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 3316–3324, 2015. (Cited on page 98.)

P

[PIT+16] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Björn Andres, Mykhaylo Andriluka, Peter Vincent Gehler, and Bernt Schiele. DeepCut: Joint subset partition and labeling for multi person pose estimation. *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 4929–4937, 2016. (Cited on pages 56, 58, 59, 67, 70, and 87.)

[PJA+12] Leonid Pishchulin, Arjun Jain, Mykhaylo Andriluka, Thorsten Thormählen, and Bernt Schiele. Articulated people detection and pose estimation: Reshaping the future. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 3178–3185. IEEE Press, 2012. (Cited on pages 7, 96, and 97.)

[PJW+11] Leonid Pishchulin, Arjun Jain, Christian Wojek, Mykhaylo Andriluka, Thorsten Thormählen, and Bernt Schiele. Learning people detection models from few training samples. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011. (Cited on pages 96 and 97.)

[PKD+16] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 2536–2544, 2016. (Cited on pages 98 and 109.)

[PMPHB17] Gerard Pons-Moll, Sergi Pujades, Sonny Hu, and Michael J. Black. ClothCap: Seamless 4D Clothing Capture and Retargeting. *ACM Trans. Graph. - Proc. ACM SIGGRAPH*, 36(4), 2017. (Cited on pages 121 and 122.)

[PMRMB15] Gerard Pons-Moll, Javier Romero, Naureen Mahmood, and Michael J. Black. Dyna: A model of dynamic human shape in motion. *ACM Trans. Graph. - Proc. ACM SIGGRAPH*, 34(4):120:1–120:14, 2015. (Cited on page 19.)

[PMTS+13] Gerard Pons-Moll, Jonathan Taylor, Jamie Shotton, Aaron Hertzmann, and Andrew Fitzgibbon. Metric Regression Forests for Human Pose Estimation. In *Proc. Br. Mach. Vis. Conf.* BMVA Press, sep 2013. (Cited on page 97.)

[PMTS+15] Gerard Pons-Moll, Jonathan Taylor, Jamie Shotton, Aaron Hertzmann, and Andrew Fitzgibbon. Metric regression forests for correspondence estimation. *Int. J. Comput. Vis.*, 2015. (Cited on page 79.)

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, 2011. (Cited on page 40.)

R

[RBD+02] Kathleen M. Robinette, Sherri Blackwell, Hein Daanen, Mark Boehmer, Scott Fleming, Tina Brill, David Hoeferlin, and Dennis Burnsides. Civilian American and European Surface Anthropometric Resource (CAESAR) final report. Technical report, US Air Force Laboratory, 2002. (Cited on page 19.)

[Rén61] Alfréd Rényi. On measures of information and entropy. In *Proc. fourth Berkeley Symp. Math. Stat. Probab.*, pages 547–561, 1961. (Cited on page 25.)

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *ArXiv e-prints*, 2015, 1505.04597. (Cited on page 98.)

[RKB04] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph. - Proc. ACM SIGGRAPH*, 23(3), 2004. (Cited on pages 62, 63, 65, and 66.)

[RKS12] Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Reconstructing 3D human pose from 2D image landmarks. In *Proc. Eur. Conf. Comput. Vis.*, pages 573–586, 2012. (Cited on pages 7, 59, and 78.)

[RKS+14] Lorenz Rogge, Felix Klose, Michael Stengel, Martin Eisemann, and Marcus Magnor. Garment Replacement in Monocular Video Sequences. *ACM Trans. Graph. - Proc. ACM SIGGRAPH*, 34(1):6:1–6:10, nov 2014. (Cited on page 97.)

[Ros58] Frank Rosenblatt. The perceptron - a probabilistic model for information storage and organization in the brain. *Psychol. Rev.*, 65, 1958. (Cited on page 15.)

[Ros12] Matt Rosenzweig. Lp Spaces for $0 < p < 1$. `https://matthewhr.files.wordpress.com/2012/09/lp-spaces-for-p-in-01.pdf`, 2012. (Cited on page 129.)

[RPZ13] Brandon Rothrock, Seyoung Park, and Song-Chun Zhu. Integrating Grammar and Segmentation for Human Pose Estimation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 3214–3221, 2013. (Cited on page 63.)

[RS16] Gregory Rogez and Cordelia Schmid. MoCap-guided Data Augmentation for 3D Pose Estimation in the Wild. In *Adv. Neural Inf. Process. Syst.*, pages 3108–3116, 2016. (Cited on page 97.)

[RTB17] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied Hands: Modeling and Capturing Hands and Bodies Together. *ACM Trans. Graph. - Proc. ACM SIGGRAPH Asia*, 36(6):245:1–245:17, 2017. (Cited on page 122.)

S

[SBB08] Leonid Sigal, Alexandru O. Balan, and Michael J. Black. Combined discriminative and generative articulated pose and non-rigid shape estimation. *Adv. Neural Inf. Process. Syst.*, pages 1337–1344, 2008. (Cited on pages 7 and 62.)

[SBB10] Leonid Sigal, Alexandru O. Balan, and Michael J. Black. HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *Int. J. Comput. Vis.*, 2010. (Cited on pages 55, 58, 64, 67, 68, 69, and 78.)

[SDBR14] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for Simplicity: The All Convolutional Net. *ArXiv e-prints*, 2014, 1412.6806. (Cited on page 51.)

[SGdA+10] Carsten Stoll, Jürgen Gall, Edilson de Aguiar, Sebastian Thrun, and Christian Theobalt. Video-based Reconstruction of Animatable Human Characters. *ACM Trans. Graph. - Proc. ACM SIGGRAPH*, 29(6):139:1–139:10, 2010. (Cited on pages 7 and 97.)

[SGF+13] Jamie Shotton, Ross Girshick, Andrew Fitzgibbon, Toby Sharp, Mat Cook, Mark Finocchio, Richard Moore, Pushmeet Kohli, Antonio Criminisi, Alex Kipman, and Andrew Blake. Efficient human pose estimation from single depth images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(12):2821–2840, 2013. (Cited on pages 79 and 83.)

[Shi99] Yu-Shan Shih. Families of splitting criteria for classification trees. *J. Stat. Comput.*, 9:309–315, 1999. (Cited on page 24.)

[SHM+16] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, jan 2016. (Cited on page 45.)

[SL99] Alberto Suárez and James Lutsko. Globally optimal fuzzy decision trees for classification and regression. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(12), 1999. (Cited on page 14.)

[SLA+15a] Alexander Schiendorfer, Christoph Lassner, Gerrit Anders, Wolfgang Reif, and Rainer Lienhart. Active Learning for Abstract Models of Collectives. In *Proc. Work. Self-Optimisation Org. Auton. Comput. Syst.*, 2015. (Cited on page 11.)

[SLA+15b] Alexander Schiendorfer, Christoph Lassner, Gerrit Anders, Wolfgang Reif, and Rainer Lienhart. Active Learning for Efficient Sampling of Control Models of Collectives. In *Proc. Int. Conf. Self-Adaptive Self-Organizing Syst.*, 2015. (Cited on page 11.)

[SLY15] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Adv. Neural Inf. Process. Syst.*, pages 3483–3491, 2015. (Cited on page 98.)

[SM75] Bhu Dev Sharma and Dharam P. Mittal. New nonadditive measures of entropy for discrete probability distributions. *J. Math. Sci.*, 10:28–40, 1975. (Cited on page 25.)

[SS07] Praveen Srinivasan and Jianbo Shi. Bottom-up Recognition and Parsing of the Human Body. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007. (Cited on page 63.)

[SS11] Min Sun and Silvio Savarese. Articulated Part-based Model for Joint Object Detection and Pose Estimation. In *Proc. IEEE Int. Conf. Comput. Vis.*, 2011. (Cited on page 63.)

[SSFMNU14] Edgar Simo-Serra, Sanja Fidler, Francesc Moreno-Noguer, and Raquel Urtasun. A High Performance CRF Model for Clothes Parsing. In *Proc. Asian Conf. Comput. Vis.*, 2014. (Cited on page 7.)

[SSK+13] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-Time Human Pose Recognition in Parts from Single Depth Images. *Commun. ACM*, 56(1):116–124, 2013. (Cited on page 97.)

[ST13] Benjamin Sapp and Ben Taskar. MODEC: Multimodal Decomposable Models for Human Pose Estimation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013. (Cited on page 78.)

[SW48] Claude Elwood Shannon and Warren Weaver. *The mathematical theory of communication.* Bell System Technical Journal, 1948. (Cited on page 24.)

[SWH+16] Shunsuke Saito, Lingyu Wei, Liwen Hu, Koki Nagano, and Hao Li. Photorealistic Facial Texture Inference Using Deep Neural Networks. *ArXiv e-prints*, 2016, 1612.00523. (Cited on pages 98 and 118.)

[SWL+13] Samuel Schulter, Paul Wohlhart, Christian Leistner, Amir Saffari, Peter Michael Roth, and Horst Bischof. Alternating Decision Forests. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 508–515, 2013. (Cited on page 14.)

[SWT11] Benjamin Sapp, David Weiss, and Ben Taskar. Parsing human motion with stretchable models. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1281–1288. IEEE, 2011. (Cited on page 78.)

[SZS+14] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *Proc. Int. Conf. Learn. Represent.*, 2014. (Cited on page 18.)

T

[Tay00] Camillo J. Taylor. Reconstruction of Articulated Objects from Point Correspondences in Single Uncalibrated Image. *Comput. Vis. Image Underst.*, 2000. (Cited on page 62.)

[TF10] Duan Tran and David Forsyth. Improved Human Parsing with a Full Relational Model. In *Proc. Eur. Conf. Comput. Vis.*, 2010. (Cited on page 63.)

[TGJ+15] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using Convolutional Networks. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 648–656, 2015. (Cited on page 70.)

[The16a] The Mountain 3D T-Shirts. 3D T-Shirt (Dog), unchanged. License: `https://creativecommons.org/licenses/by-sa/2.0/`. `https://flic.kr/p/a4c26z`, 2016. (Cited on page 9.)

[The16b] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *ArXiv e-prints*, 2016, 1605.02688. (Cited on page 46.)

[TOHC15] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a Next-Generation Open Source Framework for Deep Learning. In *Proc. Work. Mach. Learn. Syst. NIPS*, 2015. (Cited on page 47.)

[Tsa88] Constantino Tsallis. Possible generalization of Boltzmann-Gibbs statistics. *J. Stat. Phys.*, 52(1-2):479–487, 1988. (Cited on page 25.)

[TSSF12] Jonathan Taylor, Jamie Shotton, Toby Sharp, and Andrew Fitzgibbon. The Vitruvian Manifold: Inferring Dense Correspondences for One-Shot Human Pose Estimation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 103–110. IEEE, 2012. (Cited on page 97.)

V

[vdOKE+16] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, and Others. Conditional image generation with pixelcnn decoders. In *Adv. Neural Inf. Process. Syst.*, pages 4790–4798, 2016. (Cited on page 98.)

[vdOKK16] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *ArXiv e-prints*, 2016, 1601.06759. (Cited on page 98.)

[VLM+14] David Vázquez, Antonio M. Lopez, Javier Marin, Daniel Ponsa, and David Geronimo. Virtual and real world adaptation for pedestrian detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(4):797–809, 2014. (Cited on page 96.)

[vMBD+15] Bart van Merriënboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, and Yoshua Bengio. Blocks and Fuel: Frameworks for deep learning. *ArXiv e-prints*, 2015, 1506.00619. (Cited on page 46.)

[VRM+17] Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from Synthetic Humans. *ArXiv e-prints*, 2017, 1701.01370. (Cited on pages 96, 97, and 107.)

W

[WK11] Huayan Wang and Daphne Koller. Multi-level inference by relaxed dual decomposition for human pose segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 2433–2440. IEEE, 2011. (Cited on page 63.)

[WOR11] Huamin Wang, James F. O'Brien, and Ravi Ramamoorthi. Data-Driven Elastic Models for Cloth: Modeling and Measurement. *ACM Trans. Graph. - Proc. ACM SIGGRAPH*, 30(4):71:1–11, 2011. (Cited on pages 7 and 97.)

[WRKS16] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional Pose Machines. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016. (Cited on pages 7, 51, 78, 85, and 99.)

[WTL11] Yang Wang, Duan Tran, and Zicheng Liao. Learning Hierarchical Poselets for Human Parsing. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011. (Cited on page 63.)

X

[XLS+11] Feng Xu, Yebin Liu, Carsten Stoll, James Tompkin, Gaurav Bharaj, Qionghai Dai, Hans-Peter Seidel, Jan Kautz, and Christian Theobalt. Video-based Characters: Creating New Human Performances from a Multi-view Video Database. *ACM Trans. Graph. - Proc. ACM SIGGRAPH*, 30(4):32:1–32:10, 2011. (Cited on page 97.)

Y

[Yam04] Takuya Yamano. Does the Lesche condition for stability validate generalized entropies? *Phys. Lett. A*, 329(4–5):268–276, 2004. (Cited on page 27.)

[YKOB12] Kota Yamaguchi, M. Hadi Kiapour, Luis E. Ortiz, and Tamara L. Berg. Parsing Clothing in Fashion Photographs. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 3570–3577, 2012. (Cited on pages 7, 64, 67, 68, 69, 135, and 136.)

[YKOB15] Kota Yamaguchi, M. Hadi Kiapour, Luis E. Ortiz, and Tamara L. Berg. Retrieving Similar Styles to Parse Clothing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(5):1028–1040, 2015. (Cited on page 68.)

[YLL+16] Chao Yang, Xin Lu, Zhe Lin, Eli Shechtman, Oliver Wang, and Hao Li. High-Resolution Image Inpainting using Multi-Scale Neural Patch Synthesis. *ArXiv e-prints*, 2016, 1611.09969. (Cited on page 98.)

[YLO+17] Wei Yang, Shuang Li, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. Learning Feature Pyramids for Human Pose Estimation. In *Proc. Int. Conf. Comput. Vis.*, pages 1281–1290, 2017. (Cited on page 121.)

[YR11] Yi Yang and Deva Ramanan. Articulated Pose Estimation using Flexible Mixtures of Parts. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 3546–3553, 2011. (Cited on page 63.)

[YR13] Yi Yang and Deva Ramanan. Articulated Human Detection with Flexible Mixtures of Parts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(12):2878–2890, 2013. (Cited on page 7.)

[YYSL16] Xinchen Yan, Jimei Yang, Kihyuk Sohn, and Honglak Lee. Attribute2image: Conditional image generation from visual attributes. In *Proc. Eur. Conf. Comput. Vis.*, pages 776–791. Springer, 2016. (Cited on page 98.)

[YZS+15] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop. *ArXiv e-prints*, 2015, 1506.03365. (Cited on page 107.)

Z

[ZFL+10] Shizhe Zhou, Hongbu Fu, Ligang Liu, Daniel Cohen-Or, and Xiaoguang Han. Parametric Reshaping of Human Bodies in Images. *ACM Trans. Graph. - Proc. ACM SIGGRAPH*, 29(4), 2010. (Cited on pages 7, 62, 78, 79, and 97.)

[ZK16] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. *ArXiv e-prints*, 2016, 1605.07146. (Cited on page 18.)

[ZTF11] Matthew D. Zeiler, Graham W. Taylor, and Rob Fergus. Adaptive Deconvolutional Networks for Mid and High Level Feature Learning. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 2018–2025. IEEE, 2011. (Cited on page 104.)

[ZZLD15] Xiaowei Zhou, Menglong Zhu, Spyridon Leonardos, and Kostas Daniilidis. Sparse Representation for 3D Shape Estimation: A Convex Relaxation Approach. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 4447–4455, 2015. (Cited on pages 7, 59, and 88.)